# GRAPH THEORETIC TECHNIQUES FOR FACILITIES LAYOUT

---

A thesis

submitted in partial fulfilment

of the requirements for the Degree

of

Doctor of Philosophy in Operations Research

in the

University of Canterbury

by

J.W. Giffin

---

University of Canterbury

1984

CONTENTS

CONTENTS ctd....

# LIST OF FIGURES

List of Figures ctd....

List of Figures ctd....

# LIST OF TABLES

List of Tables ctd....

ACKNOWLEDGEMENTS

During the development and preparation of this dissertation a number of people have assisted in many ways. In particular, to the following I owe a special debt of gratitude.

- To my supervisor, Les Foulds, for his enthusiasm, encouragement, inspiration and advice, and for introducing me to the topic of this thesis.

- To Hans Daellenbach, John George and Don McNickle, for fostering my interest in Operations Research and for providing a unique environment in which to work. To Don, also, for his very efficient proof-reading.

- To Fred Baird and John Buchanan, for their support (in common sufferance).

- To my parents, for their patience.

- To the University Grants Committee, for financial assistance.

- To Karilyn Smith, for her expert typing in the face of illegibility.

- And, especially, to Ken Henry, for just being himself.

Thank you all.

## ABSTRACT

In this thesis we consider a two-phase graph theoretic approach to designing the layout of a system of physical facilities. Heuristic techniques are required because the complexity of the problem gives little hope for optimization. The initial phase involves determining the relative spatial adjacency of facilities in the plane. Several new formulations are developed. The basic method of maximizing the sum of pairwise adjacency scores is extended to account for near adjacency. Facility areas are then included using the more realistic objective of minimizing total transportation cost in the layout under an approximation to rectilinear travel. Considerable computational experience is given.

The ultimate aim of facilities design is the production of a scale block plan - the second phase. We present a method for systematically producing such a plan for a restricted class of adjacency graphs that is concise enough to be implemented on a microcomputer. Proposed modifications to this rationale are then outlined in the context of multifloor layout.

Efficiency is an important criterion throughout the work. Our methods use a constructional, rather than the more common improvement, rationale, and hence advantage may be taken of this updating nature. We introduce several updating schemes in order to make promising techniques tractible on problems of practical size. Included is a

modified methodology for general graph planarity testing in an updating framework.

Graph theory offers a flexible modelling base within which we may readily encapsulate many formulation alternatives. We feel that this thesis contains an important contribution in providing methods which give high quality solutions to problems unsolvable by other means in reasonable computing time. Also, out of this work are spawned several suggestions for worthwhile further development and implementation.

# CHAPTER I

# INTRODUCTION

## 1.1 The Purpose of This Study

In this study we explore aspects of the graph theoretic approach to the problem of the layout of facilities, as defined formally in Section 1.3. Facilities layout is an important problem in industrial engineering, where it is concerned mainly with designing the layout of a system of physical facilities such as buildings on a plane site or machines on a workshop floor. However, application is not restricted to the industrial context - buildings such as hospitals, libraries, universities and office blocks can also be designed within the framework.

The task is to produce a scale plan depicting the relative positions of the facilities which optimize some measure of system performance, such as total transportation cost within the system, or the sum of pairwise adjacency desirability ratings. The method of attack is to decompose the problem into two phases: the determination of the adjacency structure of the layout, and the construction of the block plan corresponding to this structure. Both problems turn out to be difficult in the sense that optimal solutions to them are not available for examples of practical size. We therefore investigate several approximation techniques for the first phase, and develop a method of obtaining a scale block plan for some restricted cases that is

implementable using microcomputer technology. Also considered is a possible rationale for extending the techniques to the design of multi-floor layouts.

Throughout the thesis the central theme is algorithmic graph theory. Though not a classical tool of Operations Research, the spirit of the technique is traditional - it aids the decision-maker in modelling and finding an acceptable solution to a real-world problem.

## 1.2 Terminology and Notation

In this section we introduce the graph theoretic notation and terminology to be used throughout the thesis (Definitions 1.1 through 1.22) and define other relevant concepts.

Definition 1.1 An undirected graph, G, is a pair of sets (V,E) where

(i) $V \neq \phi$

(ii) $|V| < \infty$, where $|V|$ is the cardinality of V

(iii) the elements of V are called vertices

(iv) the elements of E are distinct undirected pairs of vertices of V, called edges. Each edge in E is denoted by $(v_i,v_j)$.

Definition 1.2 A weighted undirected graph is an ordered pair (G,w) where

(i) the function $w : E \rightarrow \mathbb{R}$ defines edge weights

(ii) $w(v_i,v_j)$ is written $w_{ij}$ or $w_{v_i,v_j}$

Remark 1.3 Throughout the thesis we will generally regard the modifiers "weighted, undirected" as being implicit.

Definition 1.4 If $(v_i,v_j)$ is an edge of a graph we say that

(i) $v_i$ and $v_j$ are adjacent

(ii) $v_i$ and $v_j$ are each incident with $(v_i,v_j)$

Definition 1.5 The adjacency matrix $A = (a_{ij})$ of a graph is defined by

$$a_{ij} = \begin{cases} 1 \text{ if vertices } v_i \text{ and } v_j \text{ are adjacent} \\ 0 \text{ otherwise} \end{cases}$$

Definition 1.6 The degree (or valency) of a vertex $v_i$ is the number of edges incident with $v_i$, denoted $\deg(v_i)$.

Definition 1.7 If $V' \subseteq V$ and $E' \subseteq E$ then $(V',E')$ is a subgraph of G if it satisfies the properties of a graph itself.

Definition 1.8 A graph G with n vertices is said to be complete, denoted $K_n$, if all pairs of its vertices are joined by an edge. $K_n$ has $\frac{1}{2}n(n-1)$ edges.

Definition 1.9 A graph G is termed bipartite if $|V| \leq 1$ or if there exists a partition $V = [V_1|V_2]$ such that every edge in G joins a vertex of $V_1$ to a vertex of $V_2$.

Definition 1.10 A path between vertices s and t of a graph G is a sequence of edges $(s,i_1),(i_1,i_2),\ldots,(i_k,t)$. This path may be denoted an (s,t)-path.

Definition 1.11 A cycle is an (s,s) path containing at least one edge, in which no vertex except s is repeated. If every vertex in the graph is represented in the cycle, it is called a Hamiltonian cycle.

Definition 1.12 Two vertices $v_i$ and $v_j$ of a graph G are said to be connected if there exists a path between $v_i$ and $v_j$. G is connected if all pairs of vertices are connected. A component of a graph G is a maximal connected subgraph i.e.: it is not a subgraph of any other connected subgraph of G. (Hence G is connected iff it has exactly one component.)

Definition 1.13 An edge-contraction in a graph is obtained by removing an edge e (incident with vertices $v_i$ and $v_j$, say) and identifying $v_i$ and $v_j$ in such a way that the resulting vertex is incident to those edges (other than e) which were originally incident to $v_i$ or $v_j$. A contraction of a graph is then a graph which results from G after a sequence of edge-contractions.

Definition 1.14 Two graphs are homeomorphic if they can both be obtained from the same graph by inserting new vertices of degree 2 into its edges.

Definition 1.15 Two graphs $G_1$ and $G_2$ are isomorphic if there is a one-to-one correspondence between the vertices of $G_1$ and those of $G_2$ with the property that the number of edges joining any two vertices of $G_1$ is equal to the number of edges joining the corresponding vertices of $G_2$.

Definition 1.16 A plane graph is a graph drawn in the plane such that no two edges (or the curves representing the edges) intersect geometrically except at a vertex to which they are both incident.

Definition 1.17 A graph is planar if it is isomorphic to a plane graph.

Theorem 1.18 A graph is planar iff it contains no subgraph homeomorphic to $K_5$, or $K_{3,3}$ (the complete bipartite graph on 2 sets of 3 vertices).

Proof: [Kuratowski (1930)]

Theorem 1.19 A graph is planar iff it contains no subgraph that is contractible to $K_5$ or $K_{3,3}$.

Proof: [Wilson (1972)]

Theorem 1.20 For a graph $G = (V,E)$ with $|V| \geq 3$ to be planar, $|E| \leq 3|V| - 6$.

Proof: [consequence of Euler (1752)]

Definition 1.21 The regions bounded by edges in a planar graph G are called the faces of G. The unbounded region is called the exterior face.

Definition 1.22 A graph G is maximal planar if no further edges may be added to G without violating its planarity. In a maximal planar graph, $G = (V,E)$ all the faces are triangular (hence the alternative terminology of triangulation), and $|E| = 3|V| - 6$.

Remark 1.23 The remaining definitions deal with some theoretical aspects of algorithms and heuristics.

Definition 1.24 A heuristic is a procedure designed to give a good quality solution to a problem instance in reasonable computational time, but does not guarantee that the solution will be optimal.

Definition 1.25 An algorithm is a solution procedure which guarantees to find the optimal solution to a problem instance (if one exists).

Definition 1.26 Let p denote a problem instance, and let H be a heuristic applied to the problem. In a graph theoretic context, let E(H,p) denote the set of edges chosen by H and E*(p) denote the edge set of maximum weight. Then the ratio

$$R_H(p) = \frac{w(E(H,p))}{w(E^*(p))}$$

is a measure of the quality of the solution produced by H (where w(s) is the edge-weight sum of the elements of s). The worst case ratio $\rho_H$ is defined to be

$$\rho_H = \inf_p (R_H(p)).$$

$\rho_H (\leq 1)$ provides a guaranteed value for the quality of the solution in comparison with the optimum, but is usually a pessimistic measure.

Definition 1.27 The size of a problem is an integral value associated with a problem instance which represents some measure of the quantity of the input data. The input data may be thought of as representable by a string of symbols from some alphabet.

Remark 1.28 In the graph-theoretic context, problem size is usually determined by the number of vertices and/or the number of edges in the graph.

Definition 1.29 The time complexity of an algorithm (or heuristic) is the solution time required expressed as a function of the problem size. The limiting behaviour of the complexity as size increases is called the asymptotic time complexity.

Definition 1.30 A function $g(n)$ is said to be $O(f(n))$ if there exists a constant $c$ such that $g(n) \leq cf(n)$ for all but some finite (possibly empty) set of non-negative values for $n$.

Remark 1.31 As an example of algorithmic complexity, if an algorithm processes inputs of size $n$ in time $cn^2$ for some constant $c$, then we say that the time complexity for that algorithm is $O(n^2)$, read "order $n^2$".

Remark 1.32 An algorithm is generally considered a practically useful solution technique to a problem only if its complexity grows polynomially with respect to the size of the input. This includes algorithms for which the asymptotic complexity is not a polynomial itself, but is bounded by a polynomial. Such algorithms are termed polynomial-time or efficient.

Definition 1.33 For an optimization problem, let $F$ denote the set of feasible solutions and $c$ represent a cost function, $c : F \to \mathbb{R}$. Assume that $F$ and $c$ are given implicitly in terms of two algorithms, $A_F$ and $A_c$. $A_F$, given a

combinatorial object f and a set S of parameters, will decide whether $f \in F$; $A_c$, given a feasible solution f, and another set of parameters Q, returns the value of c(f).

Definition 1.34 The optimization version of a combinatorial optimization problem is defined as: given representations of the parameters S and Q for the algorithms $A_F$ and $A_c$, find the optimal feasible solution.

Definition 1.35 The recognition version of a combinatorial optimization problem is: given an instance - a representation of S and Q - and an integer L, is there a feasible solution $f \in F$ such that $c(f) \leq L$ (minimization form).

Definition 1.36 Denote by $\mathcal{P}$ the class of recognition problems that can be solved by a polynomial-time algorithm.

Definition 1.37 Included in the class $\mathcal{P}$ are the problems LINEAR PROGRAMMING, MAXIMUM MATCHING IN GRAPHS, MAXIMUM FLOW IN A NETWORK, and GRAPH PLANARITY TESTING.

Definition 1.38 Let $\Sigma$ be a fixed finite alphabet and \$ be a distinguished symbol in \$ (marking the end of the problem encoding). A recognition problem A is in the class $\mathcal{NP}$ if there exists a polynomial p(n) and an algorithm A (a certificate-checking algorithm) such that: the string $x \in \Sigma$ is a yes instance of A if and only if there exists a string of symbols in $\Sigma$, denoted c(x), (the certificate), $|c(x)| \leq p(|x|)$ with the property that A, if supplied with the input x\$c(x) reaches the answer yes after at most $p(|x|)$ steps.

Remark 1.39 $\mathcal{P} \in \mathcal{NP}$

Remark 1.40 Problems in $\mathcal{NP}$ are solvable in principle by polynomial-depth backtrack search.

Remark 1.41 Examples of problems in $\mathcal{NP}$ are TRAVELLING SALESMAN PROBLEM, INTEGER LINEAR PROGRAMMING, CLIQUE DETERMINATION IN GRAPHS and the SATISFIABILITY PROBLEM (for Boolean expressions).

Definition 1.42 Let $A_1$ and $A_2$ be recognition problems. We say that $A_1$ reduces in polynomial time to $A_2$ iff there exists a polynomial-time algorithm $A_1$ for $A_1$ that uses several times as a subroutine at unit cost a (hypothetical) algorithm $A_2$ for $A_2$. $A_1$ is called a polynomial-time reduction of $A_1$ to $A_2$.

Definition 1.43 A recognition problem $A_1$ polynomially transforms to another recognition problem $A_2$ if, given any string x, we can construct a string y within polynomial (in $|x|$) time such that x is a yes instance of $A_1$ iff y is a yes instance of $A_2$.

Remark 1.44 Polynomial-time transformations may be thought of as polynomial-time reductions with just one cell of the subroutine $A_2$, exactly at the end of the algorithm for $A_1$. The remainder of the algorithm constructs y, the input to the algorithm for $A_2$.

Definition 1.45 A recognition problem $A \in \mathcal{NP}$ is said to be $\mathcal{NP}$-complete if all other problems in $\mathcal{NP}$ polynomially transform to A.

Remark 1.46 If there is an efficient algorithm for A, above, then there is an efficient algorithm for every problem in $\mathcal{NP}$.

Remark 1.47 The problem examples of Remark 1.42 are also $\mathcal{NP}$-complete.

Remark 1.48 In order to prove that a problem is $\mathcal{NP}$-complete it is necessary to show that:

(i) the problem is in $\mathcal{NP}$.

(ii) all other problems in $\mathcal{NP}$ polynomially transform to the problem.

Part (ii) is practically achieved by showing that a known $\mathcal{NP}$-complete problem is polynomially transformable to the given problem.

For a more complete coverage of graph theoretic terminology, refer to Harary (1969), Wilson (1972) or Bondy and Murty (1976). The fundamental results on $\mathcal{NP}$-completeness may be found in Cook (1971) and Karp (1972). Garey and Johnson (1979) contains a comprehensive account of all developments since 1971, plus a large number of known $\mathcal{NP}$-complete problems. For many uncovered aspects of algorithmic computation, see Aho, Hopcroft and Ullmann (1974) or Reingold, Nievergelt and Deo (1977). The excellent book of Papadimitriou and Steiglitz (1982) covers all the above material in lucid style; the treatment has strongly influenced the contents of this section.

## 1.3 Facilities Layout Problems

The first formulation of a facilities layout problem was by Koopmanns and Beckmann (1957), as the well-known Quadratic Assignment Problem (QAP). Each facility can be thought of as being composed of a number of unit subfacility modules (usually taken as the highest common factor of all the facility areas), and the planar layout site is considered to be divided up into an orthogonal grid, with each location cell having a unit module area. QAP then has the objective of determining the relative location of the subfacilities so as to minimize total transportation cost throughout the configuration, while guaranteeing that each facility is a contiguous region.

We now describe a variation upon the original formulation, due to Hillier and Connors (1966).

Let

$N = \{1,2,\ldots,n\}$ = the set of subfacilities.

$M = \{1,2,\ldots,m\}$ = the set of grid locations.

$c_{ij}$ = the cost per unit time period of assigning subfacility i to location j.

$d_{ij}$ = the distance from location i to location j (where this distance is an appropriate measure of the travel cost between locations).

$f_{ij}$ = the work flow from subfacility i to subfacility j.

$s_i$ = the set of locations to which subfacility i may be feasibly assigned.

$$a_{ijkr} = \begin{cases} f_{ij}d_{jr} & \text{if } i \neq k \text{ or } j \neq r \\ f_{ii}d_{jj} + c_{ij} & \text{if } i = k \text{ or } j = r \end{cases}$$

$$x_{ij} = \begin{cases} 1 \text{ if subfacility } i \text{ is assigned to location } j \\ 0 \text{ otherwise} \end{cases}$$

If subfacilities i and j belong to the same facility, $f_{ij}$ is set to an artificially high value to ensure adjacency. The $c_{ij}$ values are assigned prohibitively high levels when $j \notin s_i$. QAP then takes the form

$$\text{MINIMIZE} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} \sum_{r=1}^{m} a_{ijkr} x_{ij} x_{kr}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j \in N$$

(each subfacility must be located)

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i \in M$$

(at most one subfacility per location)

$$x_{ij} = 0/1, \qquad i \in M, \; j \in N$$

If $n < m$, we may introduce $(m-n)$ dummy subfacilities, with zero $c_{ij}$ and $f_{ij}$ values. Also note that the assumption of decomposability of $a_{ijkr}$ is quite restrictive, but reduces the input requirements of the original formulation from $0(n^4)$ to $0(n^2)$.

The division of each facility into an integral number of modules circumvents the implicit QAP assumption that a facility may be represented as one object. However, the cost of this can be a significant increase in computational requirements through the expansion of the size of a problem. Configurations of unacceptable irregularity may also result, and instability (in the form of cycling) can occur during solution by a standard technique (Krarup and Pruzan (1977)).

For a summary of both algorithmic and heuristic approaches to the solution of problem QAP, see Foulds (1983).

Unfortunately, problem QAP has been shown to be NP-complete (Lenstra (1976), Sahni and Gonzalez (1976)), so that optimal solutions are available for only small problems involving up to 15 subfacilities (see, for example, Lawler (1975), Bazaraa (1979), Burkard and Stratmann (1978) and Gavett and Plyter (1966)).

We now consider an alternative formulation of the Facilities Layout problem in a graph theoretic framework (see Chapter 2 for a justification of this approach). Here we identify each facility i with the vertex $v_i$ of a graph $G = (V,E,W)$. (As in problem QAP, subfacility modules are defined as required.) We assume that a relationship chart (or matrix) W is available which summarizes the desirability of siting each pair of facilities adjacently. This adjacency between facilities $v_i$ and $v_j$, say, is represented by an edge $(v_i,v_j) \in E$ in the graph, so that there are $\frac{1}{2}|V|(|V|-1)$ possible edges to choose from; the edge-weight $w_{ij}$ corresponds to the desirability rating W(i,j). Figure 1.1 illustrates these notions. Adjacent facility vertices are joined by an edge intersecting their common boundary, so that the resultant graph, including a vertex representing the exterior facility, depicts the adjacency structure of the layout. It is termed the dual graph, and is planar iff the layout itself is planar (see, for example, Seppanen and Moore (1970)). In fact, provided no facility in the layout is nested entirely within another facility, and all wall junctions involve the meeting of three facilities, the dual adjacency graph must be maximally planar.

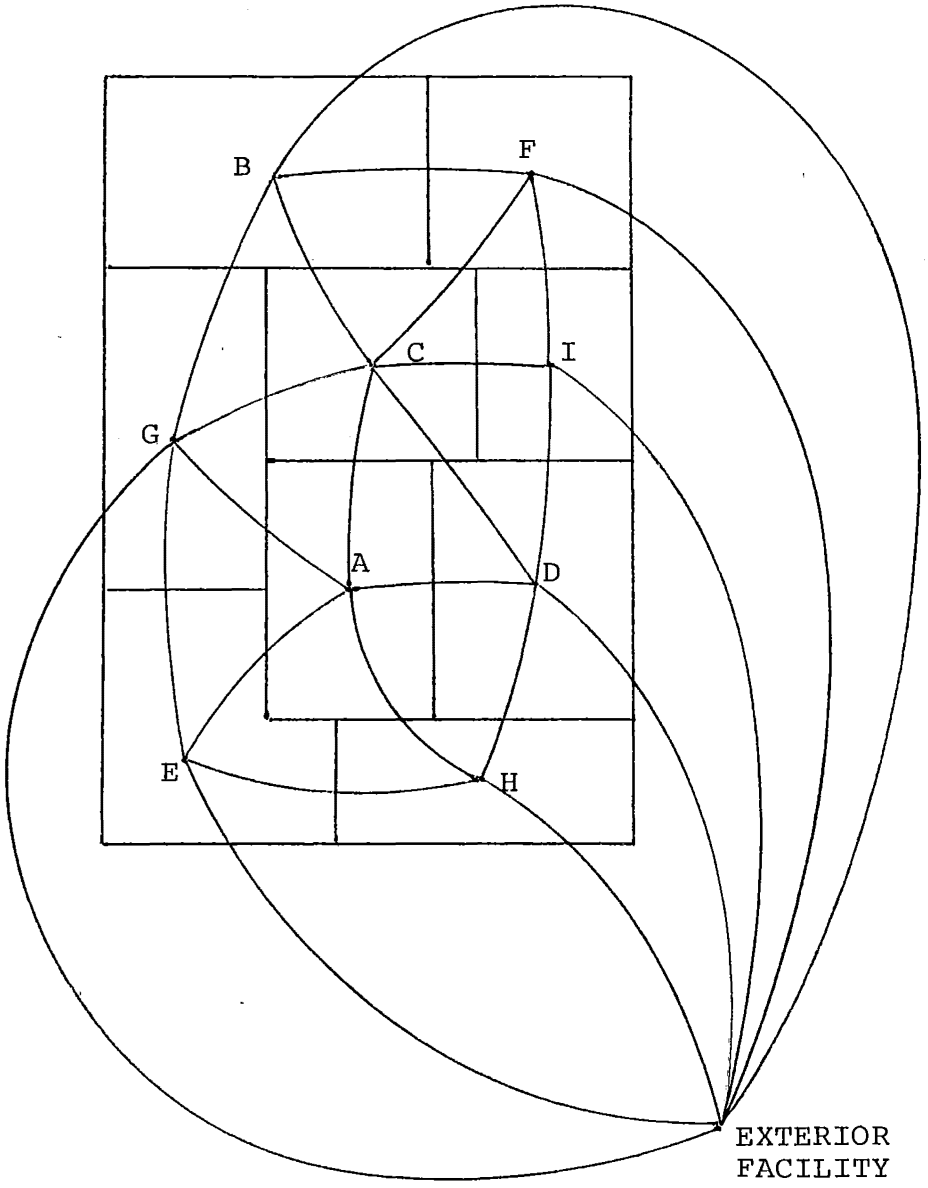



Figure 1.1 An 8-facility block plan and its maximally planar adjacency (dual) graph.

The graph theoretic formulation now takes the following form (Foulds and Robinson (1976)), which we shall call problem ADJACENCY.

Let

$w_{ij}$ = the closeness rating for siting facilities i and j adjacently.

n = the number of facilities.

V = the set of facilities.

N = the set of pairs of facilities which must be adjacent in any feasible solution.

F = the set of pairs of facilities which must not be adjacent in any feasible solution.

E = P-(N∪F) = the set of possible adjacency pairs.

$$\text{MAXIMIZE} \quad \sum_{(i,j) \in E} w_{ij} x_{ij}$$

subject to $(V, E' \cup N)$ is a maximal planar graph

$$x_{ij} = 1 \quad , \quad (i,j) \in N$$

$$x_{ij} = 0/1 \quad , \quad (i,j) \in E$$

$$x_{ij} = 0 \quad , \quad (i,j) \in F$$

where

$$x_{ij} = \begin{cases} 1 \text{ if facilities i and j are located adjacently} \\ 0 \text{ otherwise} \end{cases}$$

and

$$E' = \{(i,j) : x_{ij} = 1, \ (i,j) \in E\}$$

Hence, we have reduced the problem to that of finding a maximum weight planar subgraph of a given (complete) weighted graph.

We now show that the values $w_{ij}$ may be assumed to be greater than or equal to zero.

Theorem 1.1 In problem ADJACENCY, if we make the transformation

$$w^*_{ij} = w_{ij} + w, \text{ for all } i,j \ (w > 0)$$

then the solution set E' remains unchanged.

Proof: The modified objective function is now

$$\text{MAX} \sum_{(i,j) \in E} (w_{ij} + w)x_{ij}$$

$$= \text{MAX}\Big( \sum_{(i,j) \in E} w_{ij}x_{ij} + \sum_{(i,j) \in E} wx_{ij} \Big)$$

$$= \text{MAX}\Big( \sum_{(i,j) \in E} w_{ij}x_{ij} + w \sum_{(i,j) \in E} x_{ij} \Big)$$

$$= \text{MAX} \sum_{(i,j) \in E} w_{ij}x_{ij} + w(3n-6)$$

(since G is maximal planar)

i.e.: the objective function differs only by a constant

i.e.: the solution to the modified problem will be identical to that of problem ADJACENCY. QED

In particular, if we choose

$$w = \text{MAX}_{w_{ij} < 0} |w_{ij}| \quad ,$$

then $w^*_{ij} \geq 0$ for all i,j.

This result then implies that any optimal solution to problem ADJACENCY must correspond to a maximal planar graph, since any optimal solution that is not maximal planar may be made so by edge addition without decreasing the value of the objective function (the total edge weight).

Note that it is also sufficient to consider integer values only of $w_{ij}$, as fractional values may be scaled by the product of all the denominators. We assume integrality henceforth.

We may easily convert a minimization formulation into a maximization form, as follows.

Theorem 1.2 To convert a minimization form of problem ADJACENCY into a maximization form, it is sufficient to make the coefficient transformation

$$w^*_{ij} = L - c_{ij} \text{ for all } i,j,$$

where $L \geq \text{MAX}\{ \underset{c_{ij} \geq 0}{\text{MAX}} c_{ij}, \underset{c_{ij} < 0}{\text{MAX}} |c_{ij}| \}$

Proof: Firstly we demonstrate the equivalence of the objective functions under the indicated transformation.

$$\begin{aligned}
\text{MIN} \sum_{(i,j) \in E} c_{ij}x_{ij} &= -\text{MAX}(- \sum_{(i,j) \in E} c_{ij}x_{ij}) \\
&= -\text{MAX}( \sum_{(i,j) \in E} (-c_{ij})x_{ij}) \\
&= -\text{MAX}( \sum_{(i,j) \in E} (L-c_{ij}-L)x_{ij}) \\
&= -\text{MAX}( \sum_{(i,j) \in E} (L-c_{ij})x_{ij} - \sum_{(i,j) \in E} Lc_{ij}) \\
&= -\text{MAX}( \sum_{(i,j) \in E} (L-c_{ij})x_{ij}) + (3n-6)L
\end{aligned}$$

Therefore, the objective functions differ by only a constant.

Now we check that the solution structure is preserved. Consider the set E ordered according to non-decreasing order of weight (in the minimization case) i.e.: $E = \{\beta_1, \beta_2, \ldots, \beta_e\}$, where $e = |E|$, $\beta_k = c_{ij}$ for some i,j

and $\beta_i \leq \beta_j$ for all i,j.

Then $E^* = \{L-\beta_1, L-\beta_2, \ldots, L-\beta_e\}$

$= \{\beta^*_1, \beta^*_2, \ldots, \beta^*_e\}$, say,

will be a set with non-increasing order, with reverse ranking and the same relative weights, since

$$|\beta_i^*-\beta_j^*| = |(L-\beta_i) - (L-\beta_j)$$
$$= |\beta_j-\beta_i| = |\beta_i-\beta_j|$$

Therefore, where in the minimization context an edge $\beta_k$ may be chosen as part of the solution, in the corresponding maximization, edge $\beta_k^*$ would be chosen i.e.: the solution structure of the maximal planar subgraph will be preserved. Hence, the theorem is proved. QED

Unfortunately, like problem QAP, problem ADJACENCY is also NP-complete, as is clear from the following theorem.

Theorem 1.3 Given a graph $G = (V,E)$ and a positive integer $K \leq |E|$, the problem of determining whether there exists a subset $E' \subseteq E$ with $|E'| \geq K$ such that $G' = (V,E')$ is planar is NP-complete.

Proof: [see Liu and Geldmacher (1978), Garey and Johnson (1979)] (The problem described is a restriction of problem ADJACENCY, and the required result follows).

Hence an optimal solution may be found for only small problems; for practical instances, heuristic methods must be developed. The relative simplicity of the ADJACENCY formulation, however, makes it a more attractive proposition for such attempts.

It is worthwhile comparing the graph theoretic and QAP formulations. Problem ADJACENCY represents an adjacency prescription only - a blank layout area is assumed, with no predetermined locations specified. This allows a planner more scope for design freedom, but implies the need for a second phase in the process - the construction of the

corresponding layout. (This aspect is discussed in Chapter 7.)

The location assignment costs, $c_{ij}$, of problem QAP are assumed zero in problem ADJACENCY. This implies that the adjacency-based formulation is more suitable for the design of a new layout rather than the modification of an existing one; QAP would be effective in either situation.

No credit is given for non-adjacency in problem ADJACENCY, so that the overall cost of the layout is not explicitly considered. In fact, the implied objective may not always be valid. Simple refinements are available, however, to provide alternatives for most situations likely to be encountered.

## 1.4 The Problems to be Studied

Several aspects of the graph theoretic approach to facility layout will be discussed in the thesis.

Firstly, in Chapter 3, we describe three effective heuristics for problem ADJACENCY, including computational experience. A fourth approach is offered from a purely theoretical viewpoint.

We then consider an extension of the basic model to incorporate credit for near adjacency. Two of the original heuristics are modified and compared in Chapter 4. A biproduct of this extension is an investigation of graph planarity testing in an updating environment - characterising planarity under the addition of at most one vertex and/or edge. Chapter 5 covers the preliminary results in the context of a particular planarity testing algorithm.

In Chapter 6 we consider an alternative objective function for the problem - that of the maximization of total transportation cost in the layout, where the actual facility areas are taken into account. The required modifications to the model are presented, together with numerical results.

In Chapter 7 we propose a method for converting the output from special cases of the methods of Chapter 3 or Chapter 6 into the corresponding block plan. Practical implementation on a microcomputer is shown to be potentially effective and tractible for the given restrictions.

Finally, application of a graph theoretic approach to multi-floor layout is discussed in Chapter 8. Little comparable work is available, so computational experience is limited to model validation.

Now, to follow in Chapter 2, we give a brief overview of previous work into the basic graph theoretic model and discuss the appropriateness of such an approach.

# CHAPTER 2

# SELECTION OF A SOLUTION PROCEDURE

## 2.1 Previous use of graph theoretic methods in facilities layout and Architectural Planning

Related graph theoretic literature falls into three categories:

(i) Heuristics for determining a highly-weighted adjacency graph from a given (complete) graph (corresponding to a relationship matrix).

(ii) Algorithms for determining a maximally weighted adjacency graph from a given (complete) graph (corresponding to a relationship matrix).

(iii) Enumeration and ornamentation layout configurations.

(i) One of the first papers to appear using graph theory in the context of facilities layout was due to Levin (1964). His method for problem ADJACENCY is essentially a greedy heuristic, similar in spirit to one to be described in Chapter 3. This was followed by the "RUGR" procedure of Krejcirik (1969). Little computational experience is available for either method.

From 1970 to 1976, a series of papers by Seppanen and Moore (1970), (1975), Moore and Carrie (1976) and Moore (1976) introduced and developed a heuristic approach for problem ADJACENCY based on symbolic processing

theory, culminating in Carrie, Moore, Roczniak and Seppanen (1978). They offer two basic approaches. One is to incrementally construct a highly-weighted adjacency graph while utilising string-grammars to represent the corresponding planar embedding. The grammatical rules used circumvent the need for planarity testing, allowing only the generation of planar layout topologies, and reduce the graphical manipulation requirements to character string transformations. Four different edge-selection criteria were tested, with promising results; however, the computational effort required is large. Again, little computational experience is available.

The second approach tackles the problem via edge elimination applied to the adjacency graph, to remove "undesirable" or "incompatible" edges until a maximal planar graph is obtained. This clearly requires repeated applications of a planarity testing algorithm; the authors chose the method of Démoucron, Malgrange, and Pertuiset (1964) (see Chapters 4 and 5 for a fuller description of this algorithm, together with improvements possible for some special cases).

Hashimshony, Shaviv and Wachman (1980) use an edge elimination rationale similar to that of Seppanen and Moore (1970). Their relationship matrix differs slightly from the standard in that there are only four distinct rating values available (0,1,2,3). The process is essentially greedy, using heuristic criteria to ensure that either a minimal number of edges are deleted or only edges of smallest weight cancelled in achieving a planar adjacency graph.

Planarity testing is achieved by the procedure of Lempel, Even and Cederbaum (1966) (see also Booth and Lueker (1976) for a linear time implementation of this algorithm). Use is made of the so-called s-t numbering generated by this approach to aid in obtaining the dual graph - a process termed "unfolding". The authors also mention a second edge elimination operation, that of locating an additional vertex where two edges cross (if viable); this would lead to the introduction of a circulation area in the corresponding dual graph. Both schemes have been embedded within a decomposition-recomposition cluster-analytic layout method (Shaviv, Hashimshony and Wachman (1977), (1978)) and appear to perform satisfactorily.

(ii) Only two papers appear to have tackled the problem of finding the optimal solution to problem ADJACENCY. Foulds and Robinson (1976) employ a constructive branch-and-bound strategy based on the Kurakowski planarity criterion (Theorem 1.18). They introduce a series of filters which reduce the need for explicitly testing promising subgraphs for planarity; if the filters fail for a particular configuration, the ultimate planarity test used is the method of Hopcroft and Tarjan (1974).

Christofides, Galliani and Stefanini (1980) produce a new integer programming formulation for problem ADJACENCY. They consider a Lagrangean relaxation of the problem to derive sharp bounds used in a branch-and-bound algorithm. Branching strategies are based on both edges and faces of the graph, in terms of bound sensitivity and structural considerations. Planarity testing is again provided by

the Hopcroft and Tarjan algorithm.

Both approaches outlined can solve problems with a maximum of only 15 vertices (or facilities) in reasonable computational time. The use of exact methods, therefore, becomes restricted to providing optimal solutions to small test problems for the purpose of comparison and performance evaluation of heuristics, such as those to be presented in Chapter 3.

(iii) Several papers have appeared dealing with the counting of architectural configurations.

March and Earl (1977) demonstrate that an isomorphism exists between the set of 3-polytopes with n + 1 polygonal faces, and a class of architectural plans with n facilities, including internal circulation areas. They note that all plans with n facilities may be obtained through ornamentation from a set of so-called "fundamental" schemes corresponding to projections of convex trivalent 3-polytopes. The duals of these polytopes are also 3-polytopes with triangular faces, analogously to the dual of a maximal planar graph being trivalent. Using a trivalent polytope construction technique due to Eberhard (1891), the authors enumerate the number of potentially symmetric plans (with an orthogonal wall arrangement and at most three-way wall junctions) for up to eleven facilities.

Essentially similar methodologies may also be found in Korf (1977) and Mitchell, Steadman and Liggett (1976).

Baybars and Eastman (1980) use an operation termed "wheel-expansion" to generate the maximal planar underlying graphs of architectural arrangements (see Chapter 3 for a definition and further application of this technique in the context of weighted graphs). The concept of a pseudo-geometric dual is introduced and characterised - this creates at least one exterior face in the dual graph and allows for a specified orientation to be imposed. Deletion of edges from the adjacency graph to provide interior courtyards (in a manner similar to that of Hashimshony et al) is also investigated. A planar graph embedding routine which takes advantage of the maximal planar graph structure is given; for such a triangulation with p vertices, 2p-4 distinct embeddings are possible. Corrections to and criticisms of the paper are given in Earl (1981); he points out that the wheel-expansion operation is the dual of the "face-splitting" operation of March and Earl, essentially due to Steinitz and Rademacher (1934).

Baybars (1982) deals with the generation of floor plans with circulation spaces, again using the pseudo-geometric dual. In what is essentially an extension of Baybars and Eastman (1980), he classifies floor plans into four types according to the form taken by the interior courtyards. Each type may be generated via ornamentation - deletion of specified edges from the underlying maximal planar graph, or using a restricted form of the wheel-expansion operation. Attempts are made to accommodate architectural concerns systematically, but without dimensionality considerations.

Grason (1970) uses the "dual graph approach" to seek an "independent topological specification" of space allocation. He describes a floor plan graph with edges representing wall segments, vertices representing corners and regions representing spaces, whose dual is oriented. Planar realisation constraints are incorporated in terms of location (contiguity, accessibility), dimension (facility area, minimum dimension) and "contradiction" (adjacency restriction). Unfortunately, no computational experience with these modifications is reported. This is one of the first attempts at constructing an actual floor plan from the adjacency graph; we leave to Chapter 7 an outline of previous work within a similar vein.

## 2.2 Justification of the Graph Theoretic Approach

The work outlined in the last section indicates that Graph Theory is a useful tool in facilities layout problems. It is relatively straightforward to model the spatial relationships within the graph theoretic framework in terms of the adjacency requirements; as March and Earl (1977) pointed out, all orthogonal plans may be obtained from fundamental schemes which are the duals of triangulations.

A major advantage over the QAP formulation is the increased freedom allowed in design - rather than a prescribed grid of elemental areas, the planner has a blank area within which irregularly shaped facilities of unequal area may be manipulated. This of course implies the necessity for dividing the problem into the two phases of adjacency determination followed by development of the corresponding block plan.

Such a decomposition is also likely to allow a simplification in methodology over a "one-pass" approach, since once the adjacency information is known, the layout process is not subject to iteration or manipulation caused by infeasibilities or myopic facility placement decisions.

Provision for flexibility in objectives is desirable, and the graph theoretic formulation permits many modifications. For instance, minimization of relationship chart scores, maximization over all pairs of adjacency scores and the minimization of transportation cost for layouts are all considered within the same framework in this dissertation, with few methodological changes required in moving from one objective to another.

In the design context, the actual number of realizable architectural plans with n facilities is $0(n)$ smaller than the number of prescribable adjacency requirements. Planar graphs encapsulate this modelling specification and may be used to identify feasible interactions among facilities rather than attempting to satisfy all required ones. The introduction of edge weights then provides a mechanism by which we may objectively rank the quality of solutions obtained. Further, estimates of facility area and dimension allow a projected plan shape to be defined; at this stage, additional objective functions could well be added, based on, for example, minimal area, perimeter or construction cost.

An added advantage is the ability to provide a performance benchmark for a heuristic applied to a particular problem, in the form of an upper bound on the optimal solution value. In fact, Carrie et al (1978) predict that "the

definition of the upper bound of feasible solutions may be the most important contribution to facilities design made by Graph Theory". It is our intention in the succeeding chapters to show that it is not the only worthwhile contribution.

## 2.3 The effectiveness of computer methods for facilities design

Much discussion has appeared in the literature regarding the merits of computer-based approaches to facilities design relative to the performance of experienced planners. Several measures of problem complexity have been devised in an attempt to categorise problems as being more suitable for computer or human solution, and comparative experiments have been carried out to verify their effectiveness. Most studies have used so-called "flow-shop problems" in a QAP format, utilising codes such as CRAFT (Buffa, Armour and Vollman (1964)) and CORELAP (Lee and Moore (1967)). We now briefly chronologically review some of the relevant work done.

Hillier (1963) suggested an efficiency criterion based on a lower bound to the optimal solution, derived via calculating the cost of locating each facility individually in each location and then using an assignment algorithm to minimize the total cost of such individual assignments. Vollmann and Buffa (1966) introduced the concept of "flow dominance", essentially the coefficient of variation of the flow data. This attempts to measure the extent to which the flow pattern approaches that of an assembly line. Experience showed that problems whose flow dominance did not exceed 200% were best solved by computer.

Scriabin and Vergin (1975) conducted a "computer versus human" experiment, using CRAFT. While CRAFT was more consistent in its solution quality on the test problems of up to twenty facilities, they found that the humans' capabilities of visualising complex patterns and learning by experience gave them an advantage over sequentially-based methods, resulting in solutions up to six-percent better. In this instance, however, the humans were, perhaps unfairly, motivated by being given the results of the best CRAFT solutions beforehand. Buffa (1976) also noted that the problems used exhibited high flow dominance and involved facilities of equal area and hence were easier to solve. Unfortunately, CRAFT finds difficulty in dealing with unequal areas, due to its pairwise-exchange rationale.

Coleman (1977) criticised the experimental procedures used by Scriabin and Vergin as being biased towards the human subjects, and suggested some alternative means of appraisal, in the form of a cost-benefit analysis of human and computer performances. Time taken to reach a solution was deemed to be the costing variable. In addition, experimental results from a study using hypothetical layout problems where the subjects were never informed of the intangible and unquantified factors showed that humans' ability to account better for these factors was "idle speculation".

Cross (1977) used problems with up to 12 facilities. The computer implementation was an elementary random pairwise-exchange heuristic, using a symmetric relationship matrix of single-digit numbers as input. In this case, the

program outperformed the planners by an average of 6%, with "significantly" better results for problems involving more than eight facilities, but Cross still concluded that there were "no particular advantages to the use of computer aids in building design".

Flow dominance was observed by Block (1979) to increase with problem size. He therefore introduced upper and lower bounds on flow dominance (using a single assembly line with constant flow as a basis) to give a modified complexity rating, $c_f$, that would be unbiased. Lewis and Block (1980) concluded that a $c_f$ value of 82% represented the upper limit for layout problems whose construction could be handled by planners. Their results also showed that improvement algorithms usually gave the best solutions, outperforming FATE (Block (1978)), PLOP2 (a modified version of CRAFT) and CORELAP.

Trybus and Hopkins (1980) also found that CRAFT was superior to human designers. They questioned the validity of the 200% flow dominance breakpoint, noting that problems with very low flow dominance were easy to solve; problem size appeared to be the most important complexity criterion.

Flow dominance was again criticised in Scriabin and Vergin (1981), who noted that it is sensitive to the inclusion of one or more very large flows in the data. They also noted that Block's modified measure could be difficult to interpret on some problems, and that Hillier's efficiency criterion consistently decreased with increasing problem size, regardless of flow dominance. A suggested alternative was the idea of line-dominance, a representation of the

extent to which a problem contains assembly-line-like data. High line dominance can manifest itself in high flow dominance, so that it was not safe to conclude that visual methods have advantages in such cases. They conjectured that the presence of line dominance decreases the efficiency of both CRAFT and visual methods, but fail to provide an adequate quantifiable definition of the concept.

It can be seen in the above survey that it is generally felt that computer-based methods provide solutions at least as good as those produced by planners. Although it is difficult to make a direct cost comparison, once the developmental phase of a computer implementation is achieved, the relative speed of solution and the ability to easily generate large numbers of solutions involving different cost scenarios make computer approaches an attractive proposition for the initial phases of the layout process. We therefore feel justified in our quest for improved methods in computer-based facilities layout, to provide a useful base on which to build an efficient, practical design.

# CHAPTER 3

## MAXIMIZING THE RELATIONSHIP CHART SCORES OF ADJACENT FACILITIES

In this chapter we describe three graph theoretic heuristics for problem ADJACENCY, evaluate their efficiency and compare them computationally. Much of the work is included in Foulds, Gibbons and Giffin (1984).

### 3.1 The Deltahedron Heuristic

This method was introduced in Foulds and Robinson (1978). It constructs the required adjacency graph via a sequence of maximal planar graphs without the need for planarity testing at any stage of the process.

The initialisation step chooses four vertices from the set of n vertices representing the facilities, then joins each pair of vertices by an edge to create the graph $K_4$, the complete graph on four vertices. Three methods for achieving this configuration so that its associated weight (given by the sum of the weights of its six edges) is sufficiently high were considered during the performance evaluation of the heuristic.

```
INITIAL_1:
begin
    choose vertices i,j,k,r such that
        w_ij + w_ik + w_ir + w_jk + w_jr + w_kr is maximized
end
```

Remark 3.1 INITIAL_1 has time complexity $O(n^4)$

```
INITIAL_2:
begin
    choose vertices i,j such that
        w_ij = MAX_{p,q ∈ V} w_pq
    choose vertex k such that
        w_ij + w_jk is maximized
    choose vertex r such that
        w_ir + w_jr + w_kr is maximized
end
```

Remark 3.2 INITIAL_2 has time complexity $O(n^2)$

```
INITIAL_3:
begin
    for each vertex i ∈ V
        let M(i) = Σ_{j ∈ V} w_ij
    arrange the vertices in order of non-increasing M(i)
    choose the first four vertices as i,j,k,r
end
```

Remark 3.3 INITIAL_3 has time complexity $O(n^2)$.

INITIAL_1 creates the initial tetrahedron ($K_4$) with the highest possible weight, whereas INITIAL_2 makes a greedy choice at each stage. INITIAL_3 attempts to reflect the "average" weighting of each vertex. It turns out that the additional effort required by INITIAL_1 is not worthwhile in terms of the weight of the final maximal planar graph (MPG). In general its solution quality is inferior. Hence

we concentrate on the other two initialisation procedures.

The $K_4$ produced by any of the above procedures may be defined in the following terms:

$V(K_4) = \{i,j,k,r\}$

$E(K_4) = \{(i,j),(i,k),(i,r),(j,k),(j,r),(k,r)\}$

$T(K_4) = \{(i,j,k),(i,j,r),(i,k,r),(j,k,r)\}$,

where the set T is the set of triangles. This specification leads to the construction step, for which there are two alternatives: a fixed-order approach or a greedy approach. The fixed-order version is based on INITIAL_3 and uses the M(i) ordering. The vertices are added one at a time according to this ordering, by an insertion of a vertex into a triangle, as follows.

Suppose vertex u is inserted into triangle (a,b,c). Then

$V \leftarrow V \cup \{u\}$

$E \leftarrow E \cup \{(a,u),(b,u),(c,u)\}$

$T \leftarrow T \cup \{(a,b,u),(a,c,u),(b,c,u)\} \setminus \{(a,b,c)\}$

(Figure 3.1 illustrates the insertion process).

Each vertex in the M(i) ordering is inserted into the triangle which results in the largest weight increase. That is, if u is the vertex to be inserted, all the triangles in the present graph are examined, and the one, (a,b,c) say, maximizing $w_{au} + w_{bu} + w_{cu}$ is chosen.

In the greedy rationale, starting with the INITIAL_2 configuration, the best vertex, triangle pair is considered at each stage, i.e.: the insertion order is not predetermined.

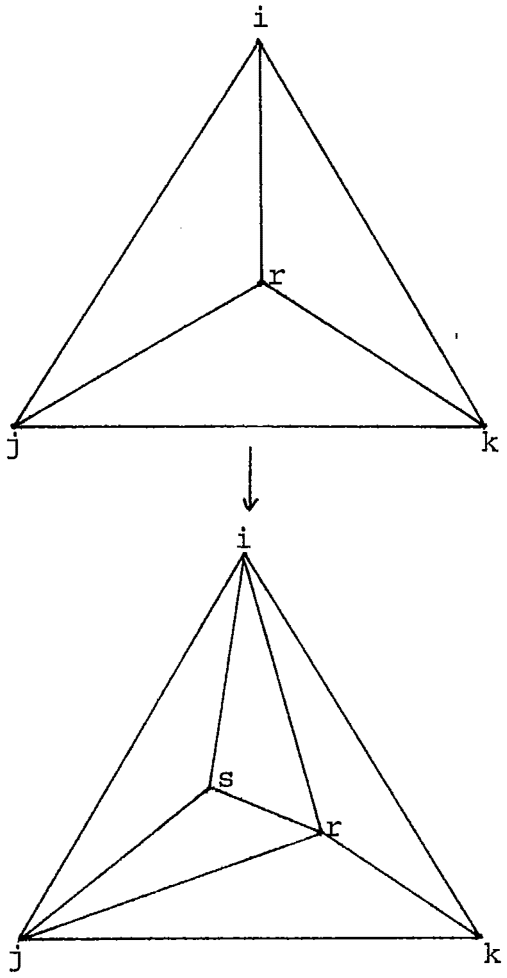



Figure 3.1 Inserting vertex s in triangle (i,j,r)

Theorem 3.1 Fixed-order DELTAHEDRON has time complexity $0(n^2)$.

Proof: INITIAL_3 is $0(n^2)$.

At the $i^{th}$ vertex insertion there are 2i+4 triangles to examine with each examination requiring constant time; and there are n-4 insertions. Hence, the time complexity is

$$0(n^2) + \sum_{i=1}^{n-4} (2i+4) = 0(n^2) + 0(n^2) = 0(n^2).$$

QED

Corollary 3.1 Greedy DELTAHEDRON has time complexity $0(n^3)$

INITIAL_2 requires $0(n^2)$ operations.

Analogously to Theorem 3.1, the time complexity is

$$0(n^2) + \sum_{i=1}^{n-4} (2i+4)(n-i)$$

$$= 0(n^2) + n \sum_{i=1}^{n-4} (2i+4) - \sum_{i=1}^{n-4} i(2i+4)$$

$$= 0(n^2) + 0(n^3) = 0(n^3).$$ QED

Considerable computational experience again shows that the greedy approach gives no advantage, so the results comparison (to be discussed later) is restricted to the fixed-order method only, using both initialisation procedures INITIAL_2 and INITIAL_3.

Figure 3.2 displays a pidgin-PASCAL description of DELTAHEDRON + INITIAL_2.

We now present two results about the theoretical performance of fixed-order DELTAHEDRON in terms of worst-case ratios.

Theorem 3.2 [Dyer, Foulds and Frieze (1983)]

(i) $\rho_{DELTAHEDRON} = 0$ ($\rho$ is defined in Definition 1.26)

(ii) let $\hat{\rho}_{DELTAHEDRON} = \inf_{Q}(R_{DELTAHEDRON}(Q))$, where the infimum is over problems Q such that $w(e) = 0/1$ for all edges e.

Then $\frac{1}{6} \leq \hat{\rho}_{DELTAHEDRON} \leq \frac{2}{9}$

Proof: For completeness we give the proof of (i).

Let $G = (V,E)$ and $w : E \to \mathbb{R}^+$.

Let m be a positive integer, and let $A_1, A_2, \ldots, A_{m+1}$ be m+1 disjoint sets of size m.

```
Procedure DELTAHEDRON
begin
    (* V(G) = vertex_set
       E(G) = edge_set
       T(G) = triangle set *)
    (* initialisation *)
    for j = 1 to n do sum(j) := 0;
    for j = 1 to n
        for i = 1 to n do
            sum(j) := sum(j) + w(i,j);
    order(*) := [ sum(i) : sum(i) ≥ sum(j) iff i < j ];
    (* set up tetrahedron *)
    V(G) := [ order(1),...,order (4)];
    E(G) := [(order(1),order(2)),...,(order(3),order(4))];
    set up T(G) corresponding to E(G);
    tot_ben :=      Σ      w(i,j)
              (i,j)∈E(G)
    for i = 5 to n do
    begin
        max_ben := -1;
        for each triangle (p,q,r) ∈ T(G) do
        begin
            ben := w(p,order(i)) + w(q,order(i)) + w(r,order(i));
            if ben > max_ben
                 then max_triangle := (p,q,r)
        end;
        V(G) := V(G) + order(i);
        E(G) := E(G) + [ (order (i),v) : V ∈ max_triangle];
        adjust T(G); (* |T(G)| = |T(G)| + 2 *)
        tot_ben := tot_ben + max_ben;
```

```
    end;
    output E(G), T(G), tot_ben;
end.
```

Figure 3.2

Let $A = \bigcup_{i=1}^{M} A_i$, $V = A \cup B$,

$B = \{b_1, b_2, \ldots, b_m\}$,

Define E as follows:

(a) an edge of weight 1 joins each pair of vertices in A

(b) an edge of weight m joins each vertex in $A_i$ to $b_i$, for i=1,...,m.

Now,

$$M(V) = \begin{cases} m^2 + m - 1 & , \quad v \in A \\ m^2 & , \quad v \in B \end{cases}$$

(where M(v) is as defined in INITIAL_3)

The insertion order is defined by the M(v) values, so that DELTAHEDRON will first choose the elements of A. The weight of the MPG at this stage will be $3m^2-6$. Then the vertices of B will be added, each insertion contributing at most 3m to the total weight. Therefore DELTAHEDRON will produce a graph $\hat{G} = (V,\hat{E})$ with $w(\hat{E}) \leq 6m^2-6$.

Now, let E* = {edges incident with B}.

G* = (V,E*) is a planar subgraph of G,

and $w(E^*) \geq m^3$.

It therefore follows that

$$\rho_{DELTAHEDRON} \leq \frac{6m^2-6}{m^3} \quad \text{for all } m > 0$$

i.e.:

$$\rho_{DELTAHEDRON} = 0.$$

QED

For the proof of (ii), refer to Dyer, Foulds and Frieze (1983).

Theorem 3.2 indicates the unfortunate result that DELTAHEDRON may perform arbitrarily badly on some (albeit pathological) examples. If restricted to zero-one problem instances, however, a positive performance guarantee is obtained. Extensive computational testing using "practical" data suggests that the heuristic performs extremely well - for details see Section 3.4.

It is interesting to note that a different definition of the vertex ordering yields an improved worst-case ratio. Suppose we redefine M(v) to be

$$M(V) = \underset{v \in E}{MAX}\ w(e) \qquad (v \in V, e \in E)$$

(if the edge weights are not distinct, and, without loss of generality $V = \{1,2,\ldots,n\}$, then perturb the weight of edge $(i,j)$ to $w(i,j) + \varepsilon^i + \varepsilon^j$), and all the resultant heuristic DELTAZ. Then

Theorem 3.3 [Dyer, Foulds, Frieze]

$$\rho_{DELTAZ} = \tfrac{1}{6} .$$

No attempt has been made to analyse the worst case performance of greedy DELTAHEDRON.

Once the final graph has been constructed by DELTAHEDRON, it can be possibly improved by one or both of the following strategies.

(i) Edge replacement

This operation essentially swaps diagonal edges within quadrilaterals in the graph. Difficulties

which arise when the other diagonal is already present may be overcome using a method described in Foulds and Robinson (1979). The edge replacement is made only if an increase in the total weight of the graph results.

(ii) Vertex relocation

A vertex of degree 3 may be removed from its insertion triangle and relocated elsewhere, again if an increase in the solution score is implied. There must always be at least one such candidate vertex - the last vertex to be inserted.

Two examples of the improvement strategies are as follows. Consider Figure 3.3 .

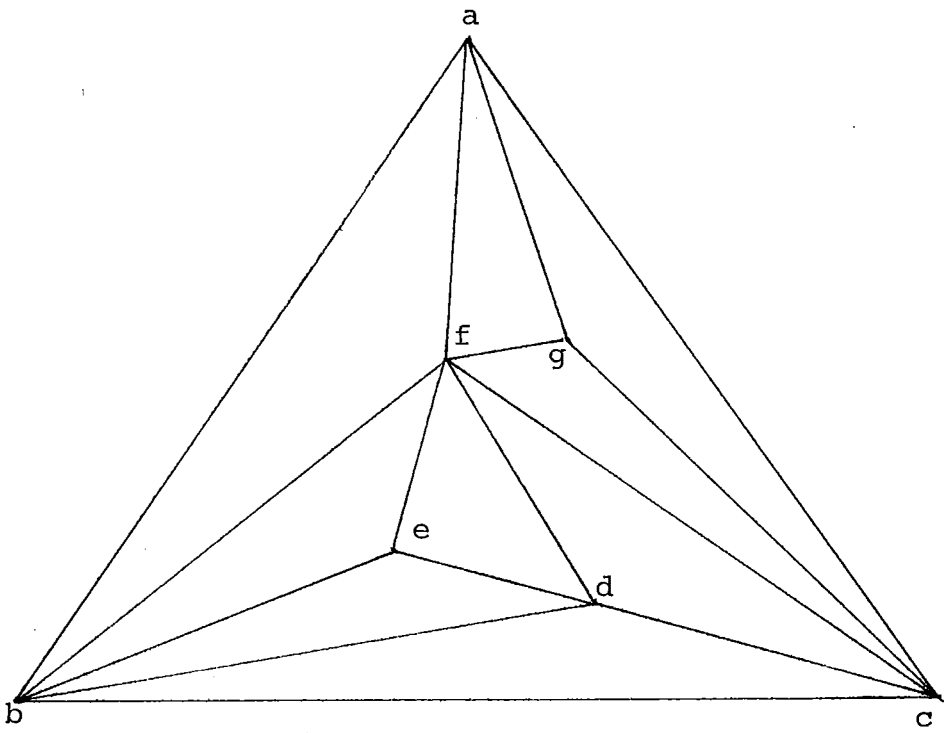



Figure 3.3

In the quadrilateral (c,d,e,f) we may replace edge (d,f) by edge (c,e) if $w_{af} - w_{ce} > 0$. This would imply the deletion of triangles (d,e,f) and c,d,f) and the addition of triangles (c,d,e) and c,e,f).

Vertex e may be relocated (unless the above edge replacement has been made) in, say, triangle (a,c,g) if $(w_{ae} + w_{ce} + w_{cg} - w_{be} - w_{de} - w_{ef}) > 0$. The triangle adjustments are straightforward.

We now consider some implementation aspects of the two improvement strategies.

Foulds and Robinson (1978) suggested constructing a list of the edges not forming part of the MPG solution in order of non-increasing weight, and checking each of these edges in a series of passes for the possibility of replacement. At the end of each pass, the replacement providing the greatest benefit (increase) is compared to the best available vertex relocation, and the better modification is performed. This process continues until no improvement is possible. Such a scheme requires $0(n^3)$ operations at each pass, but this may be reduced to $0(n^2)$ by checking only 0(n) of the $\frac{1}{2}(n-3)(n-4)$ prospective ordered edges, with an accompanying minor reduction in solution quality. However, a modification of the methodology yields an $0(n^2)$ complexity without incurring a penalty. Instead of checking whether a particular prospective edge forms the alternative diagonal of a quadrilateral in the current solution, it proves more expedient to construct a list of all the quadrilateral diagonals and hence determine a list of their feasible complements (taking note of any complements which may already occur in the original list). The maximal gain at each pass is thus determined

by the maximum difference between corresponding elements in the list, and the best improvement possible from a vertex relocation.

Both the quadrilateral list construction with the determination of the best prospective edge for insertion and the choice of the best vertex to relocate require $0(n^2)$ operations. Since, in any MPG there are 3n-6 quadrilaterals and at most n possible vertices for relocation, usually 0(n) passes are sufficient. Thus:

Remark 3.4 The improvement strategies of edge replacement and vertex relocation require $0(n^3)$ operations.

## 3.2 The Wheel Expansion Heuristic

The second heuristic for problem ADJACENCY was first introduced in Eades, Foulds and Giffin (1982). We now summarize the graph-theoretic concepts which are needed for its motivation and development.

Definition 3.1 Let G be a graph and $e \in V(G)$. Then G/e denotes the graph obtained by contracting e, in the sense of Definition 1.13 (all consequent loops are deleted, and multiple edges coalesced).

The contraction operation is illustrated in Figure 3.4.

Definition 3.2 A connected graph is called 3-connected if at least three of its vertices and their incident edges must be removed before it becomes disconnected.

part of G

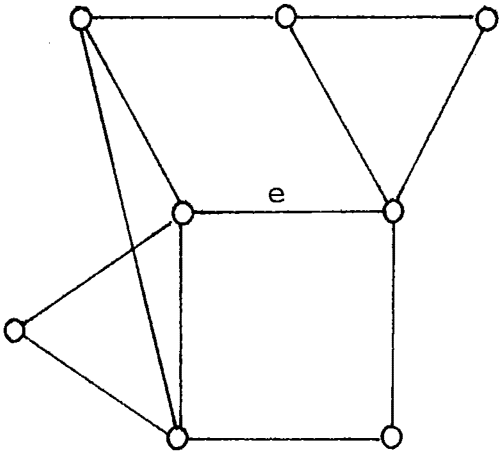



Figure 3.4 (a)

part of G/e

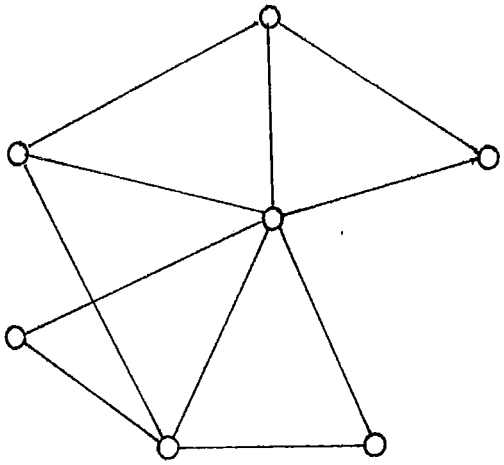


Figure 3.4 (b)

Figure 3.4 The contraction process

Theorem 3.4 [Thomassen (1980), Tutte (1961)]
If G = (V,E) is a 3-connected graph with $|V| \geq 5$, then there exists an edge $e \in E$ such that G/e is 3-connected.

This result gives a useful characterization of 3-connected graphs, and hence is applicable to maximal planar graphs (Thomassen (1980)), since contraction preserves maximal planarity. Therefore, we have the following:

Corollary 3.1 If G = (V,E) is a maximal planar graph with $|V| \geq 5$, then there exists an edge $e \in E$ such that G/e is maximal planar.

We may also define a reverse operation to contraction.

Definition 3.3 The following determines vertex-splitting in a graph G = (V,E):

Let $v \in V$ be such that $\deg(v) \geq 4$. Replace v by two adjacent vertices v' and v" so that each vertex formerly joined to v is joined to exactly one of v' and v", with $\deg(v') \geq 3$ and $\deg(v") \geq 3$ in the resulting graph.

Figure 3.5 illustrates the vertex splitting procedure.

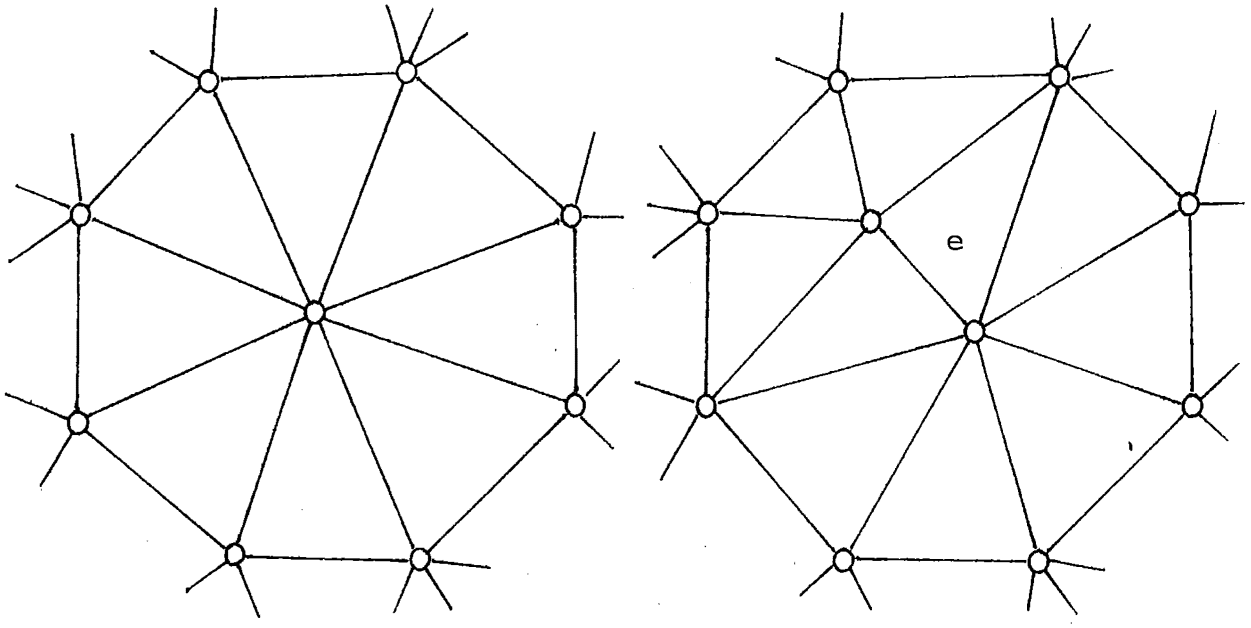



Figure 3.5 (a) Figure 3.5 (b)

Figure 3.5 An example of vertex-splitting

Definition 3.4 A wheel on n vertices is a cycle on (n-1) vertices (termed the rim), each vertex of which is adjacent to one additional vertex (termed the hub).

Definition 3.5 If G is a maximal planar graph, the application of vertex-splitting to G is called wheel expansion.

Remark 3.5 The wheel expansion operation is the same as that used in Baybars and Eastman (1980) and Baybars (1982) for constructing configurations of unweighted maximal planar graphs in an architectural context.

With this modified terminology, Corollary 3.1 becomes:

Corollary 3.2 Every maximal planar graph may be obtained from $K_4$ by a sequence of wheel expansions.

Note that this implies that wheel-expansion could be used as the basis for enumerating all maximal planar graphs on a given vertex set, whereas the vertex-insertion operation (described in procedure DELTAHEDRON) in its unimproved form may not, as it always produces a configuration containing at least one vertex of degree 3. (Note that there exist MPG's with a minimum degree of 4.) Figure 3.6 indicates how the backtracking may be done (by backtracking we mean an implicit search technique through all possible solutions).

Wheel expansion forms the basis for the heuristic to be described; firstly we give a rationale for the data structure used in the implementation.

Theorem 3.5 [Skupien (1966)]
A graph $G = (V,E)$ with $|V| = n$ and $|E| = s$ is maximal planar iff

(i) $s = 3n-6$ and

(ii) every vertex $v \in V$ is the hub of a wheel, $W_v$, which has exactly the set of vertices adjacent to v as its rim.

```
ALGORITHM MAXIMAL;
begin
Procedure backtrack(G);
begin
    if V(H) = V(G)
    then begin
        if (weight of H) > MAX
        then begin
            MAX := weight of H;
            HMAX := H;
        end;
    end
    else begin
        for each (x,y,k,r) where { x ∈ V(H)
                                 { y ∈ V(G)-V(H)
                                 { k,r ∈ W_x      } do
            backtrack (Ĥ(x,y,k,r));
    end;
end;
(* main program *)
MAX := -∞;
HMAX := (φ,φ) (* null graph *)
for each set of four vertices x1,x2,x3,x4 ∈ G do
begin
    H := planar embedding of subgraph of G induced by
                {x1,x2,x3,x4};
    backtrack(H);
end;
write out the embedding HMAX;
end.
```

Figure 3.6 An algorithm for enumerating maximal planar graphs

Theorem 3.6 [Eades, Foulds and Giffin (1982)]
For each $v \in V$, the $W_v$ of Theorem 3.5 is unique.

We choose to use this unique representation in the form of a circularly linked list, $L_v$, for each vertex, v, and create a linked list containing each of these, as shown in Figure 3.7. This is essentially a variation of the usual adjacency list representation of a graph, with the additional constraint of vertex ordering within each adjacency list.

Each MPG has a representation as defined above which (for labelled graphs) is unique to within cyclic reorderings and reversals of the lists $L_v$.

The first step of WHEEL_EXPANSION is to create the initial tetrahedron, H, using procedure INITIAL_2. At each subsequent step, a vertex is added to H by wheel expansion as follows:

Choose $x \in H$, $y \in G-H$.

Choose $k,r \in rim(W_x)$.

Expand $W_x$ to $W'_x$ and a new wheel $W_y$ so that

$k,r \in rim(W'_x)$, $k,r \in rim(W_y)$.

This is illustrated in Figure 3.8.

The choice of the vertices, x, y, k and r is made greedily, and the new graph is denoted by $\hat{H}(x,y,k,r)$.

For completeness, we now detail how the wheel expansion operation modifies the underlying data structures. Suppose that the wheels for x, k and r are represented by

$$W_x : (x_1,x_2,\ldots,x_r)$$
$$W_k : (k_1,k_2,\ldots,k_s)$$
$$W_r : (r_1,r_2,\ldots,r_t)$$

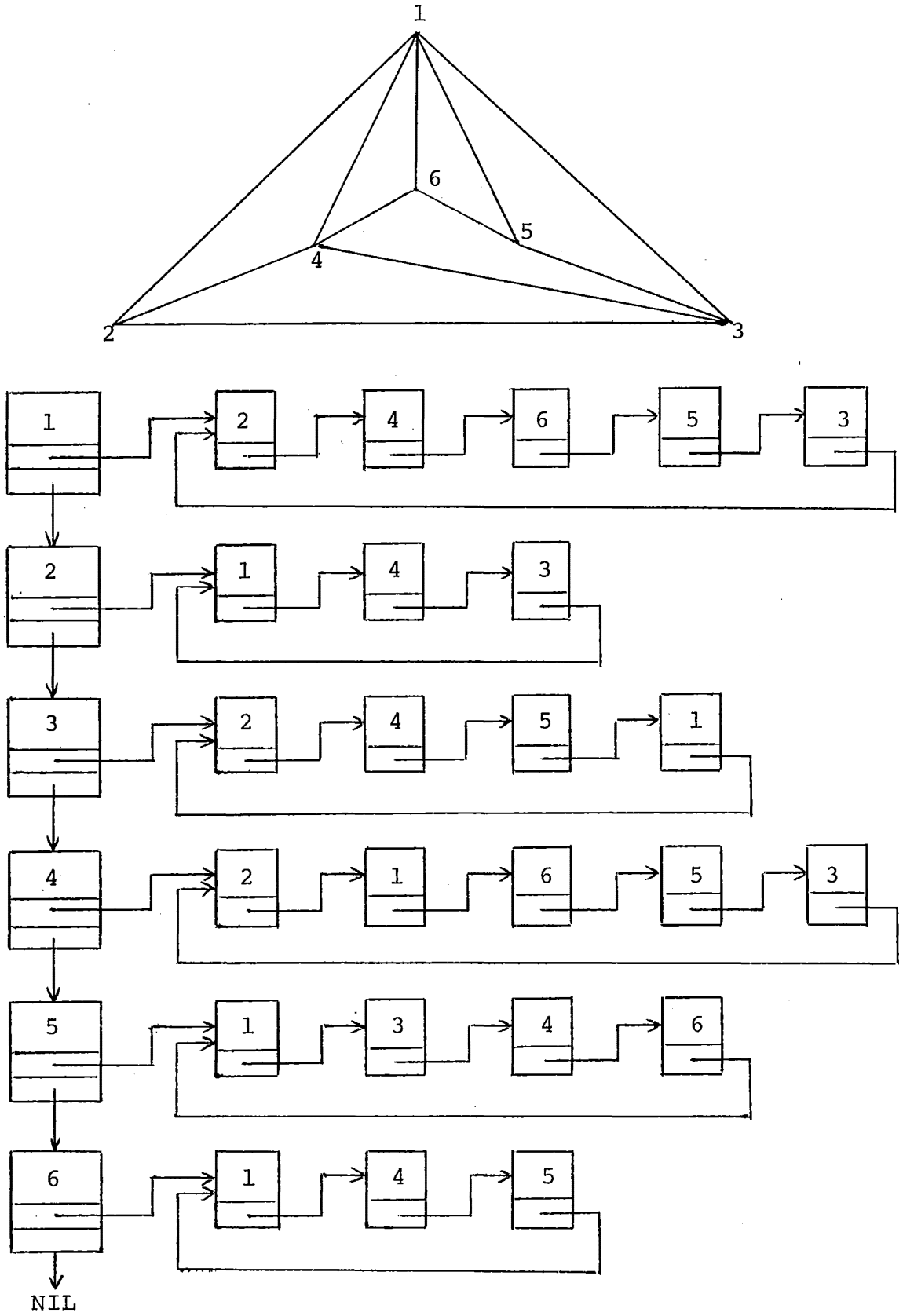



Figure 3.7 An example of an MPG and its data representation.

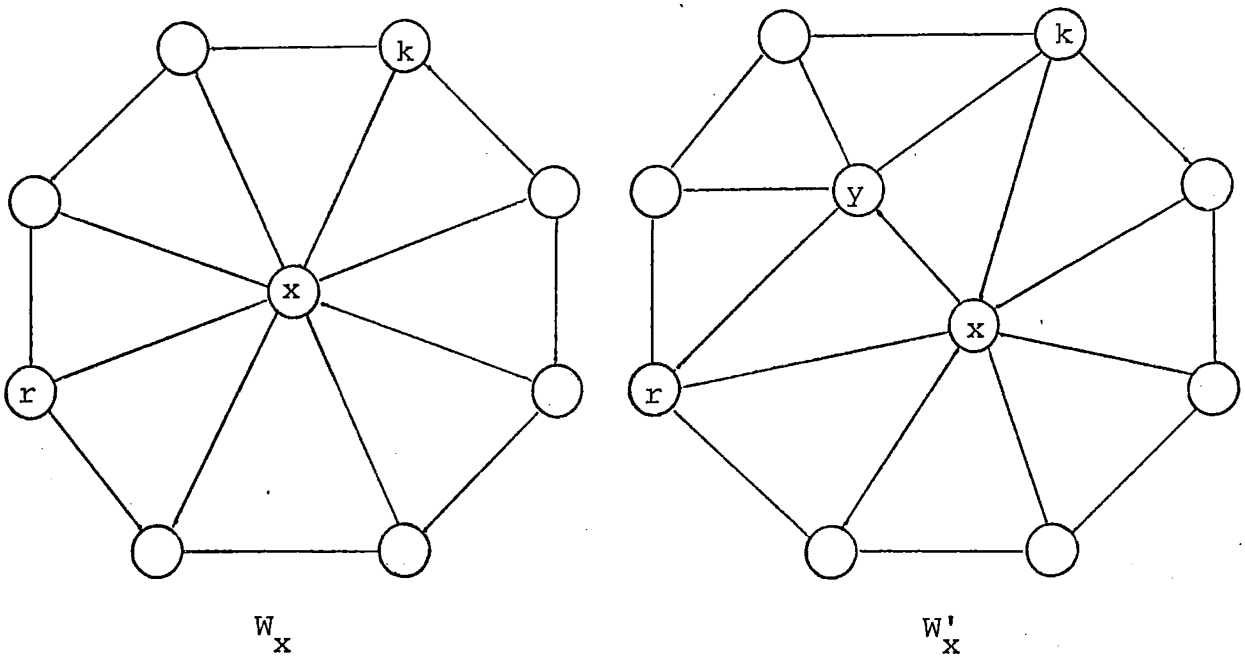



Figure 3.8 An example of wheel-expansion

Suppose that $k = x_i$ and $b = x_j$. If $i > j$ then we may reverse the order of the $W_x$ r-tuple, so we choose to assume $i < j$. To form the wheel expansion of H by y about x between k and r, we perform the following operations:

(a) The wheel about x:

$W_x = (x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_r)$

becomes

$W'_x = (x_1, x_2, \ldots, x_i, y, x_j, x_{j+1}, \ldots, x_r)$

(b) The wheel about k:

The subsequence $k_{i-1}, x, x_{i+1}$ occurs somewhere in the representation $W_k$; i.e.: for some p such that $1 \leq p \leq s$, $k_{p-1} = x_{i-1}$, $k_p = x$, $k_{p+1} = x_{i+1}$. Then

$W_k = (k_1, k_2, \ldots, k_{p-1}, k_p, k_{p+1}, \ldots, k_s)$

becomes

$$(k_1, k_2, \ldots, k_{k-1}, k_k, y, k_{k+1}, \ldots, k_s)$$

(c) The wheel about r:

Analogously to (b), the representation $W_r$ contains the subsequence $x_{j-1} = r_{q-1}$, $x = r_q$, $x_{j+1} = r_{q+1}$. Then

$W_r = (r_1, r_2, \ldots, r_{q-1}, r_q, r_{q+1}, \ldots, r_t)$

becomes

$$(r_1, r_2, \ldots, r_{q-1}, r_q, y, r_{q+1}, \ldots, r_t)$$

(d) The wheel about y:

Wheel-expansion creates this new wheel,

$W_y = (k, x, r, x_{i+1}, x_{i+2}, \ldots, x_{j-1})$.

As previously mentioned, care must be taken to recognise that these representations are equivalent under cyclic shifts and reversals. An efficient method for determining the weights of the successive embeddings of each 4-triple (x,y,k,r) during the greedy evaluation of all feasible 4-triples. Minor differences in data structure updating also exist when the vertices k and r are "separated" by one or more wheels interior to $\text{rim}(W_x)$.

Figure 3.9 briefly summarizes the WHEEL_EXPANSION heuristic.

Note that considering all possible choices of x, y, k and r results in the slowest implementation of WHEEL_EXPANSION. An alternative would be to choose one parameter (x, say) according to some greedy criterion (for example, the vertex with the heaviest wheel) and then never revise the choice.

```
Procedure WHEEL_EXPANSION;
begin
    INITIAL_2; (* greedy tetrahedron choice *)
    repeat n-4 times
        find x,y,k,r such that { x ∈ H, y ∈ G-H
                               { k,r ∈ rim(W_x)
                               { Ĥ(x,y,k,r) has maximum weight
        H := Ĥ(x,y,k,r);
    end;
    output H;
end.
```

Figure 3.9 The wheel-expansion heuristic

This would then imply the need to consider only all possible triples of the remaining parameters to determine the "best" $\hat{H}(x,y^*,a^*,b^*)$, resulting in a faster, but probably inferior, implementation.

In section 3.4, only the first variation is tested. The following theorem establishes its time complexity.

Theorem 3.7 [Eades, Foulds and Giffin (1982)]

WHEEL_EXPANSION has worst case time complexity $0(n^4)$.

Remark 3.6 When implemented on random graphs of bounded degree, the observed average running time of wheel expansion is $0(n^3)$.

Again, no attempt has yet been made to produce a worst-case ratio for WHEEL_EXPANSION because of the wealth of special cases that would have to be considered.

Note that the improvement strategies described in the last section may equally be applied to an MPG derived via WHEEL_EXPANSION.

## 3.3 The Greedy Heuristic

This heuristic is conceptually simpler than either DELTAHEDRON or WHEEL_EXPANSION. The first step of GREEDY is to order the edges in decreasing order of weight. Each edge is then examined in this order, and accepted as part of the solution being constructed unless its addition would violate the planarity of the graph. In this latter case the edge is rejected. Once a maximally planar subgraph spanning the vertex set is obtained, the heuristic is terminated. Figure 3.10 gives a pidgin-PASCAL version of GREEDY.

The planarity testing phase uses the linear-time algorithm of Hopcroft and Tarjan (1974) only after a series of filtering tests (detailed in Foulds and Robinson (1976)) have failed. This reduces the computational burden considerably.

Theorem 3.8 GREEDY has time complexity $0(n^3)$.

Proof: In the worst case, $0(n^2)$ edges must be considered. Determining whether or not the resultant graph after each edge addition is planar requires $0(n)$ operations. Hence the result.

QED

An exact worst-case performance result is also otainable for GREEDY:

Theorem 3.9 [Dyer, Foulds and Frieze (1983)]

$$\rho_{GREEDY} = \frac{1}{3}$$

Proof: Let $R = \{1,\ldots,n\}$ be a finite set and $S \subseteq P(R)$ be a system of subsets of E with the monotonicity property

$$s_1 \subseteq s_2 \in S \Rightarrow s_1 \in S.$$

Then we call (R,S) an independence system. GREEDY is actually the Greedy Heuristic (see Korte and Hausmann (1978)) applied to the independence system consisting of the sets of edges of G = (V,E) that induce planar graphs, so that the worst case occurs when w(e) = 0/1 for all $e \in E$.

If $|V| = n$, let $U_n$ be the subgraph of unit-weight edges. Then the problem becomes that of determining a maximum (edge) cardinality planar subgraph in $U_n$. If $U_n$ has c connected components, then its maximum cardinality subgraph contains at most (3n-6c) edges. GREEDY always constructs an edge maximal planar subgraph of $U_n$, which must contain a spanning tree of each of the components of $U_n$. Hence the cardinality of the heuristic subgraph is at least n-c. Therefore, the worst-case ratio is at least

$$\frac{n-c}{3n-6c} > \frac{1}{3} \quad \text{for any } c > 1. \qquad \text{QED}$$

Dyer, Foulds and Frieze (1983) also exhibit a family of graphs for which this bound is tight.

We note that GREEDY is an incremental procedure in that only one edge is being added at each stage to an already planar graph. The planarity test (as implemented) takes no account of this observation; Chapter 5 will explore a possible methodology for improving the efficiency of general planarity testing in an updating context.

```
Procedure GREEDY;
begin
    ordered_list(*) := [edges (i,j) ordered according to
                              non-increasing w(i,j)]
    G := (V(G),φ);
    k := 1;
    count := 0;
    tot_ben := 0;
    while count < 3n-6 do
    begin
        if G + ordered_list(k) PLANAR
        then begin
            G := G + ordered_list(k);
            tot_ben := tot_ben + w(ordered_list(k));
            count := count + 1;
        end;
        k := k + 1;
    end;
    output G, tot_ben;
end.
```

Figure 3.10 The Greedy heuristic

## 3.4 Performance Comparison and Evaluation

We now turn to a computational comparison of the three heuristics for problem ADJACENCY : DELTAHEDRON, WHEEL_EXPANSION and GREEDY. To determine a measure of the success of the improvement strategies, we also include IMPROVED_DELTAHEDRON, in which the edge and vertex manipulations are appended to DELTAHEDRON.

A series of random test problems was generated by using the method of Box and Muller (1958), which, for given n, $\mu$ and $\sigma$ produces normally distributed deviates as edge weights for the complete weighted graph G on n vertices with weights having mean $\mu$ and variance $\sigma^2$. Given sufficiently wide variation in $\sigma$, it is considered possible to model most of the types of problems likely to be encountered in practice, except perhaps in extreme cases of highly dominant facility relationships. Varying $\sigma$ also allows relative performance measures to be established over a range of problem types.

For $\sigma$ = 5, 10, 15, 20, 25 and 30, five problems with $\mu$ = 100 were generated for n = 10, 20 and 30, and two for n = 40. For n = 10, performance comparison with the optimal solution value (denoted by Z*) is possible; the branch-and-bound method of Foulds and Robinson was used to determine this value. For n = 20, 30 and 40, the only measure of solution quality available is the "3n-6 bound", B, mentioned in section 3.2.

Tables 3.1 to 3.9 give representative samples of the results obtained. In each case, H denotes the solution value obtained by the particular heuristic, and the symbol "*" indicates the highest solution value obtained for a particular problem. $\sigma$ increases from 5 to 30 in sets of 5. The programs for DELTAHEDRON and IMPROVED_DELTAHEDRON were written in Algol, while those for WHEEL_EXPANSION, GREEDY and the branch-and-bound routine were written in PASCAL. The standard of programming for each is considered to be at an equivalent level, with little significant difference between the languages, so that comparisons may be drawn fairly in terms of efficiency.

All tests were made on the Burroughs B6930 at the University of Canterbury.

Solution quality is considered to be the basic performance indicator, since most facility layout problems are "one-off" - the computational costs are therefore small in comparison to the possible savings available. If a large number of problems are to be considered (for example, if a wide set of scenarios is being explored), then CPU time becomes more important. Aggregated CPU times are given in Table 3.10. These times do not include times for the preparation of random graphs or for the output of results. Storage requirements are approximately $0(n^2)$ for all the heuristics, so need not be considered in relative trade-offs.

Tables 3.1 to 3.4 clearly demonstrate the superiority of DELTAHEDRON over WHEEL_EXPANSION in terms of solution quality for most problems. Table 3.10 also shows that the smaller order of time complexity for DELTAHEDRON leads to substantially lower processing time requirements (by a factor of more than thirty for n = 40). Procedure INITIAL_3 proved more effective than INITIAL_2 for DELTAHEDRON, but no clear-cut choice can be made for WHEEL_EXPANSION.

IMPROVED_DELTAHEDRON and GREEDY provide a more difficult comparison. For higher variance problems, GREEDY produces a slightly higher solution value, but at the expense of considerably longer execution time. For most smaller problems, either method appears equally likely to achieve the better quality solution. Note that for the n = 10 problem set, the optimal solution was attained by one or other of the heuristics for half the examples. We may conjecture that

TABLE 3.1: COMPARISON OF DELTAHEDRON AND WHEEL_EXPANSION FOR n = 10

| PROBLEM | σ | DELTAHEDRON | | | | WHEEL_EXPANSION | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | INITIAL_3 | | INITIAL_2 | | INITIAL_3 | | INITIAL_2 | |
| | | H | $\frac{100H}{z^*}$ | H | $\frac{100H}{z^*}$ | H | $\frac{100H}{z^*}$ | H | $\frac{100H}{z^*}$ |
| 1 | 5 | 2479* | 99.8 | 2472 | 99.5 | 2467 | 99.3 | 2467 | 99.3 |
| 2 | 5 | 2460* | 99.9 | 2457 | 99.8 | 2454 | 99.7 | 2454 | 99.7 |
| 3 | 5 | 2446 | 99.4 | 2439 | 99.1 | 2448* | 99.5 | 2448 | 99.5 |
| 4 | 5 | 2492 | 99.6 | 2484 | 99.3 | 2487 | 99.4 | 2476 | 99.0 |
| 5 | 5 | 2524* | 99.8 | 2518 | 99.6 | 2521 | 99.7 | 2512 | 99.3 |
| 6 | 10 | 2533 | 98.7 | 2535 | 98.8 | 2546* | 99.2 | 2535 | 98.8 |
| 7 | 10 | 2508 | 98.2 | 2517* | 98.6 | 2505 | 98.1 | 2508 | 98.2 |
| 8 | 10 | 2517* | 99.1 | 2511 | 98.8 | 2499 | 98.3 | 2499 | 98.3 |
| 9 | 10 | 2595* | 99.2 | 2587 | 98.9 | 2569 | 98.2 | 2587 | 98.9 |
| 10 | 10 | 2553* | 99.8 | 2542 | 99.4 | 2530 | 98.9 | 2530 | 98.9 |
| 11 | 15 | 2705 | 99.3 | 2670 | 98.0 | 2709* | 99.4 | 2680 | 98.3 |
| 12 | 15 | 2580* | 99.1 | 2597 | 99.8 | 2541 | 97.6 | 2574 | 98.9 |
| 13 | 15 | 2580* | 99.0 | 2573 | 98.8 | 2553 | 98.0 | 2559 | 98.2 |
| 14 | 15 | 2623* | 98.9 | 2623 | 98.9 | 2620 | 98.8 | 2620 | 98.8 |
| 15 | 15 | 2654 | 98.5 | 2645 | 98.2 | 2671* | 99.1 | 2661 | 98.8 |
| 16 | 20 | 2851 | 99.5 | 2864* | 100.0 | 2776 | 96.9 | 2776 | 96.9 |
| 17 | 20 | 2745* | 99.5 | 2710 | 98.2 | 2743 | 99.4 | 2743 | 99.4 |
| 18 | 20 | 2675* | 99.8 | 2652 | 99.0 | 2638 | 98.4 | 2638 | 98.4 |
| 19 | 20 | 2704* | 98.0 | 2667 | 96.6 | 2665 | 96.6 | 2702 | 97.9 |
| 20 | 20 | 2671 | 98.7 | 2623 | 97.0 | 2656 | 98.2 | 2672* | 98.8 |
| 21 | 25 | 2805* | 98.1 | 2734 | 95.6 | 2792 | 97.7 | 2792 | 97.7 |
| 22 | 25 | 2763* | 99.3 | 2728 | 98.0 | 2706 | 97.2 | 2727 | 98.0 |
| 23 | 25 | 2600* | 99.3 | 2552 | 97.5 | 2545 | 97.2 | 2600* | 99.3 |
| 24 | 25 | 2813* | 98.3 | 2809 | 98.2 | 2773 | 96.9 | 2769 | 96.8 |
| 25 | 25 | 2831* | 99.8 | 2788 | 98.2 | 2725 | 96.0 | 2725 | 96.0 |
| 26 | 30 | 3012* | 99.0 | 2893 | 95.1 | 2975 | 97.8 | 2942 | 96.7 |
| 27 | 30 | 2674* | 99.1 | 2638 | 97.8 | 2589 | 96.0 | 2654 | 98.4 |
| 28 | 30 | 2965 | 97.7 | 2894 | 95.4 | 2911 | 95.9 | 3004* | 99.0 |
| 29 | 30 | 2797* | 98.5 | 2752 | 96.9 | 2750 | 96.8 | 2746 | 96.7 |
| 30 | 30 | 3023* | 99.7 | 3011 | 99.3 | 2991 | 98.6 | 2959 | 97.6 |
| | TOTALS: | 80178 | | 79485 | | 79355 | | 79559 | |

TABLE 3.2: COMPARISON OF DELTAHEDRON AND WHEEL_EXPANSION FOR n = 20

| PROBLEM | σ | DELTAHEDRON INITIAL_3 H | DELTAHEDRON INITIAL_3 100H/B | DELTAHEDRON INITIAL_2 H | DELTAHEDRON INITIAL_2 100H/B | WHEEL_EXPANSION INITIAL_3 H | WHEEL_EXPANSION INITIAL_3 100H/B | WHEEL_EXPANSION INITIAL_2 H | WHEEL_EXPANSION INITIAL_2 100H/B |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 5686* | 98.8 | 5680 | 98.7 | 5653 | 98.2 | 5664 | 98.4 |
| 2 | 5 | 5655* | 98.6 | 5653 | 98.6 | 5621 | 98.0 | 5632 | 98.2 |
| 3 | 5 | 5654* | 98.5 | 5613 | 97.8 | 5617 | 97.8 | 5618 | 97.8 |
| 4 | 5 | 5619 | 98.5 | 5600 | 98.2 | 5630* | 98.7 | 5630* | 98.7 |
| 5 | 5 | 5630* | 98.3 | 5586 | 97.5 | 5594 | 97.6 | 5597 | 97.7 |
| 6 | 10 | 5925* | 97.1 | 5923 | 97.1 | 5896 | 96.6 | 5882 | 96.4 |
| 7 | 10 | 5828 | 96.8 | 5821 | 96.7 | 5846 | 97.1 | 5854* | 97.3 |
| 8 | 10 | 5963 | 97.3 | 5975* | 97.5 | 5969 | 97.4 | 5950 | 97.1 |
| 9 | 10 | 5845 | 96.9 | 5874* | 97.4 | 5811 | 96.3 | 5837 | 96.8 |
| 10 | 10 | 5872 | 96.7 | 5834 | 96.0 | 5892* | 97.0 | 5871 | 96.7 |
| 11 | 15 | 6172 | 94.4 | 6180 | 94.5 | 6255 | 95.7 | 6282* | 96.1 |
| 12 | 15 | 6318 | 96.5 | 6331* | 96.7 | 6231 | 95.1 | 6257 | 95.5 |
| 13 | 15 | 6006* | 95.3 | 5919 | 93.9 | 5957 | 94.5 | 5970 | 94.7 |
| 14 | 15 | 6072* | 96.2 | 5976 | 94.7 | 5992 | 95.0 | 6004 | 95.2 |
| 15 | 15 | 6119 | 95.7 | 6108 | 95.6 | 6141* | 96.1 | 6141* | 96.1 |
| 16 | 20 | 6138 | 95.3 | 6107 | 94.8 | 6091 | 94.5 | 6116 | 94.9 |
| 17 | 20 | 6314* | 94.2 | 6214 | 92.7 | 6172 | 92.1 | 6248 | 93.2 |
| 18 | 20 | 6234* | 94.2 | 6231 | 94.2 | 6204 | 93.7 | 6195 | 93.6 |
| 19 | 20 | 6358* | 94.7 | 6326 | 94.2 | 6216 | 92.6 | 6209 | 92.5 |
| 20 | 20 | 6452* | 96.4 | 6411 | 95.8 | 6350 | 94.9 | 6372 | 95.2 |
| 21 | 25 | 6817 | 94.4 | 6704 | 92.9 | 6853* | 94.9 | 6764 | 93.7 |
| 22 | 25 | 6657* | 93.7 | 6553 | 92.2 | 6597 | 92.9 | 6558 | 92.3 |
| 23 | 25 | 6501* | 92.6 | 6470 | 92.2 | 6493 | 92.5 | 6475 | 92.3 |
| 24 | 25 | 6416 | 92.3 | 6424* | 92.4 | 6307 | 90.7 | 6340 | 91.2 |
| 25 | 25 | 6608 | 92.9 | 6574 | 92.5 | 6580 | 92.5 | 6628* | 93.2 |
| 26 | 30 | 6886* | 93.5 | 6555 | 89.0 | 6796 | 92.3 | 6852 | 93.0 |
| 27 | 30 | 6894* | 92.8 | 6686 | 90.0 | 6750 | 90.8 | 6886 | 92.7 |
| 28 | 30 | 6457* | 92.0 | 6329 | 90.2 | 6361 | 90.7 | 6354 | 90.6 |
| 29 | 30 | 7002 | 92.8 | 6830 | 90.5 | 7060* | 93.6 | 6924 | 91.8 |
| 30 | 30 | 6831 | 93.3 | 6836* | 93.4 | 6680 | 91.3 | 6834 | 93.4 |
| | TOTALS: | 186929 | | 185423 | | 185615 | | 185944 | |

TABLE 3.3: COMPARISON OF DELTAHEDRON AND WHEEL_EXPANSION FOR n = 30

| PROBLEM | σ | DELTAHEDRON INITIAL_3 H | DELTAHEDRON INITIAL_3 100H/B | DELTAHEDRON INITIAL_2 H | DELTAHEDRON INITIAL_2 100H/B | WHEEL_EXPANSION INITIAL_3 H | WHEEL_EXPANSION INITIAL_3 100H/B | WHEEL_EXPANSION INITIAL_2 H | WHEEL_EXPANSION INITIAL_2 100H/B |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 8822 | 98.0 | 8826* | 98.1 | 8806 | 97.8 | 8773 | 97.5 |
| 2 | 5 | 8818* | 98.2 | 8798 | 97.9 | 8774 | 97.7 | 8774 | 97.7 |
| 3 | 5 | 8812 | 97.6 | 8822* | 97.8 | 8794 | 97.4 | 8802 | 97.5 |
| 4 | 5 | 8820* | 98.2 | 8792 | 98.0 | 8801 | 98.1 | 8775 | 97.8 |
| 5 | 10 | 8792* | 97.8 | 8756 | 97.4 | 9792* | 97.8 | 8780 | 97.7 |
| 6 | 10 | 9255* | 96.8 | 9229 | 96.5 | 9175 | 96.0 | 9191 | 96.1 |
| 7 | 10 | 9182* | 96.5 | 9114 | 95.7 | 9182* | 96.5 | 9129 | 95.9 |
| 8 | 10 | 9284* | 95.9 | 9261 | 95.7 | 9216 | 95.0 | 9236 | 95.2 |
| 9 | 10 | 9249* | 96.2 | 9235 | 96.0 | 9202 | 95.7 | 9210 | 95.8 |
| 10 | 10 | 9200 | 96.3 | 9158 | 95.8 | 9132 | 95.5 | 9215* | 96.4 |
| 11 | 15 | 9745 | 94.4 | 9795* | 95.2 | 9761 | 94.9 | 9676 | 94.0 |
| 12 | 15 | 9679 | 94.8 | 9704* | 95.0 | 9608 | 94.0 | 9639 | 94.4 |
| 13 | 15 | 9725 | 95.1 | 9578 | 93.7 | 9606 | 94.0 | 9642 | 94.3 |
| 14 | 15 | 9732* | 95.1 | 9687 | 94.6 | 9542 | 93.2 | 9540 | 93.2 |
| 15 | 15 | 9564* | 94.1 | 9525 | 93.7 | 9293 | 91.4 | 9424 | 92.7 |
| 16 | 20 | 10086* | 93.9 | 9946 | 92.4 | 9926 | 92.2 | 9902 | 92.0 |
| 17 | 20 | 9908* | 93.9 | 9678 | 91.7 | 9853 | 93.3 | 9770 | 92.6 |
| 18 | 20 | 10121* | 94.5 | 9983 | 93.4 | 9947 | 93.0 | 9886 | 92.4 |
| 19 | 20 | 10103* | 94.2 | 9967 | 93.0 | 9933 | 92.7 | 9824 | 91.7 |
| 20 | 20 | 10058 | 94.1 | 9960 | 93.5 | 9849 | 92.4 | 9976 | 93.6 |
| 21 | 25 | 10460 | 91.5 | 10555* | 92.3 | 10546 | 92.2 | 10533 | 92.1 |
| 22 | 25 | 10130* | 92.1 | 9959 | 90.6 | 9984 | 90.8 | 10103 | 91.9 |
| 23 | 25 | 10468 | 92.8 | 10262 | 90.9 | 10481* | 92.9 | 10253 | 90.9 |
| 24 | 25 | 10579 | 92.6 | 10593* | 92.7 | 10508 | 92.0 | 10456 | 91.5 |
| 25 | 25 | 10263 | 91.7 | 10373* | 92.7 | 10063 | 89.9 | 10156 | 90.7 |
| 26 | 30 | 10825* | 93.2 | 10626 | 91.5 | 10566 | 91.0 | 10587 | 91.2 |
| 27 | 30 | 10892 | 89.7 | 10824 | 89.2 | 10664 | 87.8 | 10900* | 89.8 |
| 28 | 30 | 10769 | 91.1 | 10661 | 90.2 | 10729 | 90.8 | 10821* | 91.5 |
| 29 | 30 | 10930 | 90.9 | 10426 | 90.9 | 10878 | 90.5 | 10943* | 91.1 |
| 30 | 30 | 11008 | 91.9 | 10461 | 87.3 | 10718 | 89.4 | 11034* | 92.1 |
| | TOTALS: | 295279 | | 292554 | | 293329 | | 292950 | |

TABLE 3.4: COMPARISON OF DELTAHEDRON AND WHEEL_EXPANSION FOR n = 40

| PROBLEM | σ | DELTAHEDRON | | | | WHEEL_EXPANSION | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | INITIAL_3 | | INITIAL_2 | | INITIAL_3 | | INITIAL_2 | |
| | | H | $\frac{100H}{B}$ | H | $\frac{100H}{B}$ | H | $\frac{100H}{B}$ | H | $\frac{100H}{B}$ |
| 1 | 5 | 12057* | 98.0 | 12039 | 97.9 | 12017 | 97.7 | 11999 | 97.5 |
| 2 | 5 | 12019* | 97.6 | 12002 | 97.5 | 11989 | 97.4 | 12007 | 97.5 |
| 3 | 10 | 12648* | 96.2 | 12600 | 95.8 | 12439 | 94.6 | 12487 | 95.0 |
| 4 | 10 | 12758* | 95.9 | 12736 | 95.7 | 12630 | 94.9 | 12664 | 95.2 |
| 5 | 15 | 13107 | 93.1 | 13103 | 93.1 | 12919 | 91.8 | 13112* | 93.1 |
| 6 | 15 | 13275* | 93.7 | 13240 | 93.4 | 13190 | 93.1 | 13145 | 92.8 |
| 7 | 20 | 13716 | 91.7 | 13849* | 92.6 | 13676 | 91.4 | 13506 | 90.3 |
| 8 | 20 | 13839* | 92.5 | 13705 | 91.6 | 13561 | 90.6 | 13640 | 91.1 |
| 9 | 25 | 14409* | 91.7 | 14292 | 91.0 | 14134 | 89.9 | 14171 | 90.2 |
| 10 | 25 | 14412 | 91.8 | 14500* | 92.4 | 14215 | 90.6 | 13988 | 89.1 |
| 11 | 30 | 15428* | 90.5 | 15136 | 88.8 | 15063 | 88.4 | 15232 | 89.4 |
| 12 | 30 | 15127* | 90.1 | 15052 | 89.7 | 14820 | 88.3 | 14822 | 88.3 |
| | TOTALS: | 162795 | | 162254 | | 160653 | | 160773 | |

TABLE 3.5: Z* VERSUS B FOR n = 10

| PROBLEM | $\sigma$ | Z* | $\frac{100Z^*}{B}$ | B |
|---|---|---|---|---|
| 1 | 5 | 2484 | 99.6 | 2493 |
| 2 | 5 | 2463 | 99.6 | 2472 |
| 3 | 5 | 2460 | 99.7 | 2468 |
| 4 | 5 | 2501 | 99.6 | 2512 |
| 5 | 5 | 2529 | 99.5 | 2541 |
| 6 | 10 | 2567 | 99.2 | 2587 |
| 7 | 10 | 2554 | 99.4 | 2569 |
| 8 | 10 | 2541 | 99.3 | 2558 |
| 9 | 10 | 2617 | 98.7 | 2651 |
| 10 | 10 | 2558 | 99.5 | 2572 |
| 11 | 15 | 2725 | 99.5 | 2740 |
| 12 | 15 | 2603 | 98.4 | 2644 |
| 13 | 15 | 2605 | 98.6 | 2643 |
| 14 | 15 | 2651 | 98.4 | 2695 |
| 15 | 15 | 2694 | 99.3 | 2713 |
| 16 | 20 | 2864 | 98.7 | 2901 |
| 17 | 20 | 2759 | 99.2 | 2780 |
| 18 | 20 | 2680 | 98.5 | 2722 |
| 19 | 20 | 2760 | 98.8 | 2793 |
| 20 | 20 | 2705 | 98.8 | 2738 |
| 21 | 25 | 2859 | 99.4 | 2876 |
| 22 | 25 | 2783 | 98.4 | 2827 |
| 23 | 25 | 2618 | 97.8 | 2677 |
| 24 | 25 | 2861 | 97.9 | 2923 |
| 25 | 25 | 2838 | 99.0 | 2868 |
| 26 | 30 | 3041 | 98.9 | 3075 |
| 27 | 30 | 2698 | 98.2 | 2748 |
| 28 | 30 | 3035 | 96.3 | 3151 |
| 29 | 30 | 2840 | 97.4 | 2917 |
| 30 | 30 | 3032 | 98.0 | 3093 |

TABLE 3.6: COMPARISON OF IMPROVED_DELTAHEDRON AND GREEDY, n = 10

| PROBLEM | σ | IMPROVED_DELTAHEDRON INITIAL_3 H | IMPROVED_DELTAHEDRON INITIAL_3 100H/Z* | IMPROVED_DELTAHEDRON INITIAL_2 H | IMPROVED_DELTAHEDRON INITIAL_2 100H/Z* | GREEDY H | GREEDY 100H/Z* |
|---|---|---|---|---|---|---|---|
| 1 | 5 | 2482 | 99.9 | 2472 | 99.5 | 2483* | 99.9 |
| 2 | 5 | 2460* | 99.9 | 2460* | 99.9 | 2450 | 99.5 |
| 3 | 5 | 2455* | 99.8 | 2447 | 99.5 | 2454 | 99.8 |
| 4 | 5 | 2498* | 99.9 | 2487 | 99.4 | 2492 | 99.6 |
| 5 | 5 | 2524 | 99.8 | 2529* | 100.0 | 2525 | 99.8 |
| 6 | 10 | 2541 | 99.0 | 2540 | 98.9 | 2559* | 99.7 |
| 7 | 10 | 2508 | 98.2 | 2529 | 99.0 | 2554* | 100.0 |
| 8 | 10 | 2541* | 100.0 | 2511 | 98.8 | 2541* | 100.0 |
| 9 | 10 | 2597 | 99.2 | 2617* | 100.0 | 2600 | 99.0 |
| 10 | 10 | 2556 | 99.9 | 2556 | 99.9 | 2558* | 100.0 |
| 11 | 15 | 2715 | 99.6 | 2720 | 99.8 | 2725* | 100.0 |
| 12 | 15 | 2580 | 99.1 | 2598* | 99.8 | 2582 | 99.2 |
| 13 | 15 | 2580 | 99.0 | 2600* | 99.8 | 2593 | 99.5 |
| 14 | 15 | 2631 | 99.2 | 2641 | 99.6 | 2647* | 99.8 |
| 15 | 15 | 2671 | 99.1 | 2678 | 99.4 | 2694* | 100.0 |
| 16 | 20 | 2851 | 99.5 | 2864* | 100.0 | 2819 | 98.4 |
| 17 | 20 | 2752 | 99.7 | 2757 | 99.9 | 2757* | 99.9 |
| 18 | 20 | 2680* | 100.0 | 2652 | 99.0 | 2657 | 99.1 |
| 19 | 20 | 2715 | 98.4 | 2667 | 96.6 | 2751* | 99.7 |
| 20 | 20 | 2694* | 99.6 | 2681 | 99.1 | 2693 | 99.6 |
| 21 | 25 | 2803 | 98.0 | 2818 | 98.6 | 2859* | 100.0 |
| 22 | 25 | 2763* | 99.3 | 2737 | 98.3 | 2747 | 98.7 |
| 23 | 25 | 2600* | 99.3 | 2614 | 99.8 | 2600* | 99.3 |
| 24 | 25 | 2839* | 99.2 | 2819 | 98.5 | 2834 | 99.1 |
| 25 | 25 | 2838* | 100.0 | 2794 | 98.4 | 2838* | 100.0 |
| 26 | 30 | 3041* | 100.0 | 2926 | 96.2 | 3026 | 99.5 |
| 27 | 30 | 2674 | 99.1 | 2680* | 99.3 | 2669 | 98.9 |
| 28 | 30 | 3012 | 99.2 | 2976 | 98.1 | 3035* | 100.0 |
| 29 | 30 | 2820 | 99.3 | 2791 | 98.3 | 2840* | 100.0 |
| 30 | 30 | 3032* | 100.0 | 3011 | 99.3 | 3005 | 99.1 |
| | TOTAL: | 80453 | | 80172 | | 80587 | |

TABLE 3.7: COMPARISON OF IMPROVED_DELTAHEDRON AND GREEDY, n = 20

| PROBLEM | σ | IMPROVED_DELTAHEDRON INITIAL_3 H | IMPROVED_DELTAHEDRON INITIAL_3 100H/B | IMPROVED_DELTAHEDRON INITIAL_2 H | IMPROVED_DELTAHEDRON INITIAL_2 100H/B | GREEDY H | GREEDY 100H/B |
|---|---|---|---|---|---|---|---|
| 1 | 5 | 5700* | 99.0 | 5700* | 99.0 | 5699 | 99.0 |
| 2 | 5 | 5670 | 98.8 | 5672 | 98.9 | 5682* | 99.1 |
| 3 | 5 | 5666 | 98.7 | 5636 | 98.2 | 5672* | 98.8 |
| 4 | 5 | 5637 | 98.8 | 5622 | 98.6 | 5648* | 99.0 |
| 5 | 5 | 5632 | 98.3 | 5618 | 98.1 | 5644* | 98.5 |
| 6 | 10 | 5979 | 98.0 | 5984* | 98.1 | 5977 | 97.9 |
| 7 | 10 | 5894* | 97.9 | 5858 | 97.3 | 5881 | 97.7 |
| 8 | 10 | 6017* | 98.2 | 5987 | 97.7 | 6002 | 97.9 |
| 9 | 10 | 5919 | 98.1 | 5918 | 98.1 | 5926* | 98.2 |
| 10 | 10 | 5919 | 97.4 | 5900 | 97.1 | 5956* | 98.1 |
| 11 | 15 | 6250 | 95.6 | 6232 | 95.3 | 6337* | 96.8 |
| 12 | 15 | 6345 | 96.9 | 6346* | 96.9 | 6342 | 96.8 |
| 13 | 15 | 6030 | 95.7 | 5998 | 95.2 | 6119* | 97.1 |
| 14 | 15 | 6096 | 96.6 | 6075 | 96.3 | 6111* | 96.9 |
| 15 | 15 | 6144 | 96.1 | 6178 | 96.7 | 6232* | 97.5 |
| 16 | 20 | 6137* | 95.2 | 6107 | 94.8 | 6080 | 94.4 |
| 17 | 20 | 6417* | 95.7 | 6253 | 93.3 | 6361 | 94.9 |
| 18 | 20 | 6365* | 96.2 | 6297 | 95.1 | 6280 | 94.9 |
| 19 | 20 | 6426 | 95.7 | 6406 | 95.4 | 6482* | 96.5 |
| 20 | 20 | 6490* | 97.0 | 6460 | 96.5 | 6489 | 96.9 |
| 21 | 25 | 6897 | 95.1 | 6832 | 94.6 | 6888* | 95.4 |
| 22 | 25 | 6878* | 96.8 | 6642 | 92.2 | 6744 | 94.9 |
| 23 | 25 | 6606 | 94.1 | 6560 | 93.5 | 6703* | 95.5 |
| 24 | 25 | 6442 | 92.6 | 6499 | 93.4 | 6561* | 94.3 |
| 25 | 25 | 6750* | 94.9 | 6644 | 93.4 | 6714 | 94.4 |
| 26 | 30 | 7011* | 95.2 | 6830 | 92.7 | 6932 | 94.1 |
| 27 | 30 | 6939 | 93.4 | 6867 | 92.4 | 7067* | 95.1 |
| 28 | 30 | 6508 | 92.7 | 6656* | 94.9 | 6581 | 93.8 |
| 29 | 30 | 7067 | 93.7 | 7021 | 93.0 | 7205* | 95.5 |
| 30 | 30 | 6850 | 93.6 | 6944 | 94.9 | 6991* | 95.5 |
| | TOTAL: | 188681 | | 187742 | | 189306 | |

TABLE 3.8: COMPARISON OF IMPROVED_DELTAHEDRON AND GREEDY, n = 30

| PROBLEM | σ | IMPROVED_DELTAHEDRON INITIAL_3 H | IMPROVED_DELTAHEDRON INITIAL_3 100H/B | IMPROVED_DELTAHEDRON INITIAL_2 H | IMPROVED_DELTAHEDRON INITIAL_2 100H/B | GREEDY H | GREEDY 100H/B |
|---|---|---|---|---|---|---|---|
| 1 | 5 | 8843* | 98.3 | 8840 | 98.2 | 8834 | 98.2 |
| 2 | 5 | 8843 | 98.4 | 8819 | 98.2 | 8864* | 98.7 |
| 3 | 5 | 8883* | 98.4 | 8836 | 97.9 | 8866 | 98.2 |
| 4 | 5 | 8835 | 98.5 | 8810 | 98.2 | 8872* | 98.9 |
| 5 | 5 | 8864* | 98.6 | 8788 | 97.8 | 8843 | 98.4 |
| 6 | 10 | 9324* | 97.5 | 9293 | 97.2 | 9275 | 97.0 |
| 7 | 10 | 9201 | 96.6 | 9153 | 96.1 | 9213* | 96.8 |
| 8 | 10 | 9355 | 96.5 | 9306 | 95.9 | 9372* | 96.6 |
| 9 | 10 | 9281 | 96.6 | 9276 | 96.5 | 9285* | 96.6 |
| 10 | 10 | 9242 | 96.7 | 9222 | 96.5 | 9257* | 96.9 |
| 11 | 15 | 9782 | 95.1 | 9848* | 95.7 | 9847 | 95.7 |
| 12 | 15 | 9742 | 95.4 | 9790* | 95.8 | 9780 | 95.7 |
| 13 | 15 | 9776 | 95.6 | 9658 | 94.5 | 9813* | 96.0 |
| 14 | 15 | 9785* | 95.6 | 9714 | 94.9 | 9761 | 95.4 |
| 15 | 15 | 9668* | 95.1 | 9590 | 94.4 | 9610 | 94.5 |
| 16 | 20 | 10172 | 94.5 | 10115 | 94.0 | 10269* | 95.4 |
| 17 | 20 | 9982* | 94.6 | 9902 | 93.8 | 9899 | 93.8 |
| 18 | 20 | 10139* | 94.8 | 10103 | 94.5 | 10124 | 94.7 |
| 19 | 20 | 10235 | 95.5 | 10016 | 93.5 | 10267* | 95.8 |
| 20 | 20 | 10165* | 95.4 | 10077 | 94.6 | 10123 | 95.0 |
| 21 | 25 | 10686 | 93.5 | 10607 | 92.8 | 10727* | 93.8 |
| 22 | 25 | 10338 | 94.0 | 10102 | 91.9 | 10449* | 95.0 |
| 23 | 25 | 10573* | 93.7 | 10494 | 93.0 | 10571 | 93.7 |
| 24 | 25 | 10677 | 93.4 | 10711* | 93.7 | 10657 | 93.3 |
| 25 | 25 | 10551 | 94.3 | 10520 | 94.0 | 10643* | 95.1 |
| 26 | 30 | 10911* | 94.0 | 10800 | 93.0 | 10761 | 92.7 |
| 27 | 30 | 11169 | 92.0 | 11136 | 91.7 | 11289* | 93.0 |
| 28 | 30 | 11029 | 93.3 | 11086 | 93.8 | 11079* | 93.7 |
| 29 | 30 | 10940 | 90.4 | 11058 | 92.0 | 11192* | 93.1 |
| 30 | 30 | 11241* | 93.8 | 10749 | 89.7 | 11225 | 93.7 |
| | TOTAL: | 298232 | | 296437 | | 298767 | |

TABLE 3.9: COMPARISON OF IMPROVED_DELTAHEDRON AND GREEDY, n = 40

| PROBLEM | σ | IMPROVED_DELTAHEDRON INITIAL_3 H | IMPROVED_DELTAHEDRON INITIAL_3 $\frac{100H}{B}$ | IMPROVED_DELTAHEDRON INITIAL_2 H | IMPROVED_DELTAHEDRON INITIAL_2 $\frac{100H}{B}$ | GREEDY H | GREEDY $\frac{100H}{B}$ |
|---|---|---|---|---|---|---|---|
| 1 | 5 | 12074 | 98.2 | 12101* | 98.4 | 12056 | 98.0 |
| 2 | 5 | 12064* | 98.0 | 12050 | 97.9 | 12017 | 97.6 |
| 3 | 10 | 12704 | 96.6 | 12715* | 96.7 | 12696 | 96.6 |
| 4 | 10 | 12811 | 96.3 | 12831 | 96.5 | 12840* | 96.5 |
| 5 | 15 | 13263 | 94.2 | 13230 | 94.0 | 13327* | 94.7 |
| 6 | 15 | 13433 | 94.8 | 13358 | 94.3 | 13442* | 94.9 |
| 7 | 20 | 13923 | 93.1 | 13966 | 93.3 | 14082* | 94.1 |
| 8 | 20 | 14002 | 93.6 | 13902 | 92.9 | 14142* | 94.5 |
| 9 | 25 | 14489 | 92.2 | 14573 | 92.7 | 14674* | 93.4 |
| 10 | 25 | 14532 | 92.6 | 14588 | 92.9 | 14670* | 93.5 |
| 11 | 30 | 15514 | 91.0 | 15465 | 90.7 | 15684* | 92.0 |
| 12 | 30 | 15300 | 91.2 | 15405* | 91.8 | 15288 | 91.1 |
| | TOTAL: | 164109 | | 164174 | | 164918 | |

TABLE 3.10: COMPUTATIONAL TIME (seconds)

| n | : | 10 | | 20 | | 30 | | 40 | |
|---|---|---|---|---|---|---|---|---|---|
| | | M+ | S+ | M | S | M | S | M | S |
| DELTAHEDRON | : | 0.15 | 0.01 | 0.38 | 0.02 | 0.72 | 0.03 | 1.13 | 0.04 |
| IMPROVEMENTS | : | | | | | | | | |
| ONE PASS | : | 0.07 | | 0.28 | | 0.71 | | 3.55 | |
| TOTAL | : | 0.17 | 0.09 | 1.82 | 1.08 | 7.37 | 3.54 | 19.29 | 5.2 |
| WHEEL_EXPANSION | : | 0.27 | 0.01 | 3.67 | 0.49 | 13.41 | 0.81 | 39.49 | 2.31 |
| GREEDY | : | 2.0 | 0.21 | 20.2 | 2.7 | 71.3 | 6.4 | 165.8 | 14.7 |
| MPS* | : | ~200 | ~300 | | | | | | |

+M and S are the mean and standard deviation (respectively) of the computational times taken over all the runs.

*Maximal Planar Subgraph (Optimum)

values very close to the optimum have been obtained for higher n, where the bound B becomes increasingly more blunt. This may be less likely at higher variance, however, where the results show a steady deterioration in quality relative to B.

An interesting observation is the value of the poorest quality solution obtained by DELTAHEDRON, at 87.3% of B (for $n = 30$, $\sigma = 30$), and GREEDY, at 91.1% of B (for $n = 40$, $\sigma = 30$). These compare very favourably with the worst-case values of Theorems 3.2 and 3.9, indicating that these results apply only to ill-behaved pathological examples rather than those likely to be encountered in practice.

In conclusion, DELTAHEDRON and WHEEL_EXPANSION are both effective heuristics for problem ADJACENCY, with DELTAHEDRON considerably more efficient and giving slightly better quality solutions. GREEDY achieves the best quality solutions over all, but the smaller computational requirements of IMPROVED_DELTAHEDRON make it perhaps the more attractive proposition if resources are limited.

A modification of GREEDY, introduced in Chapter 4, and requiring considerably less CPU-time, provides another possible alternative. If resources are not a limiting factor, running a suite of all the methods may provide the greatest benefit.

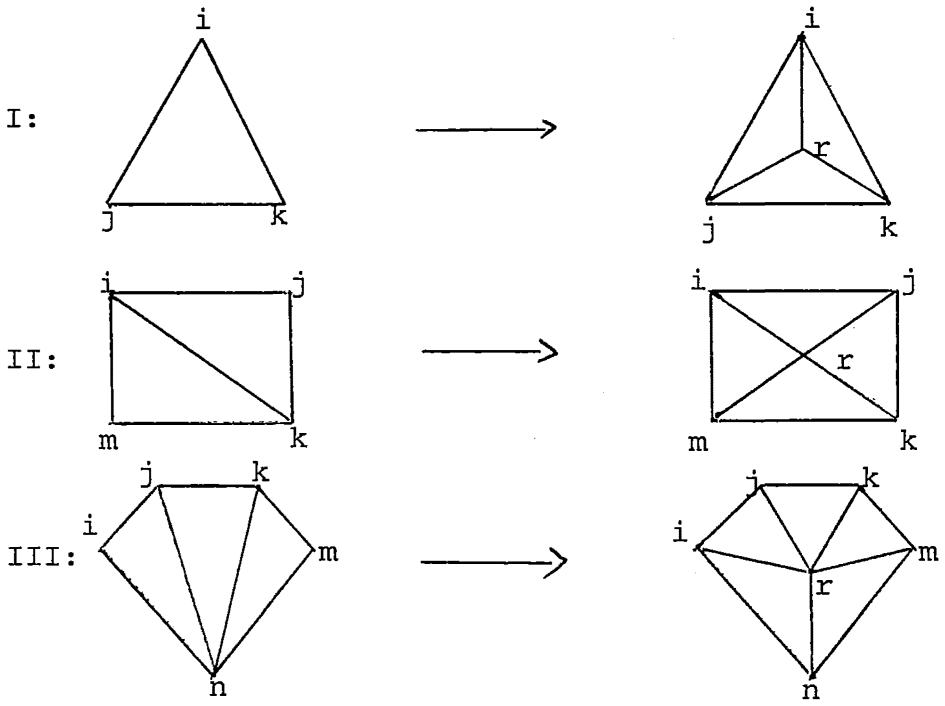
## 3.5 The WHEEL GENERATION Heuristic

In this last section we propose another heuristic for problem ADJACENCY and briefly describe a possible data structure for its implementation. Due to the success and efficiency of the other methods presented in this chapter, it was not considered worthwhile to test it computationally because of

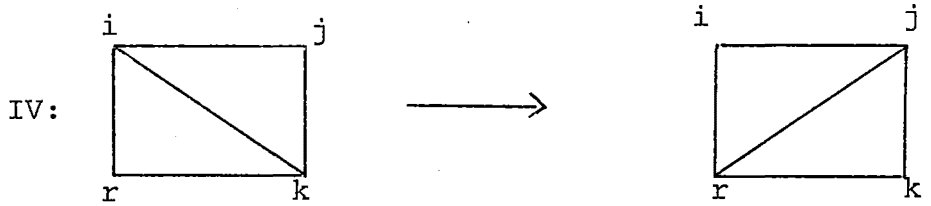
the small likelihood of it outperforming the simple but effective IMPROVED_DELTAHEDRON.

Firstly we introduce a further characterization of maximal planar graphs.

Theorem 3.10 [Bowen and Fisk (1967)]
The following three operations are sufficient to generate all maximal planar graphs from $K_4$:




Proposition 3.1 Operations II and III may be performed equivalently via application of the sequence operation I and the following operation (IV):



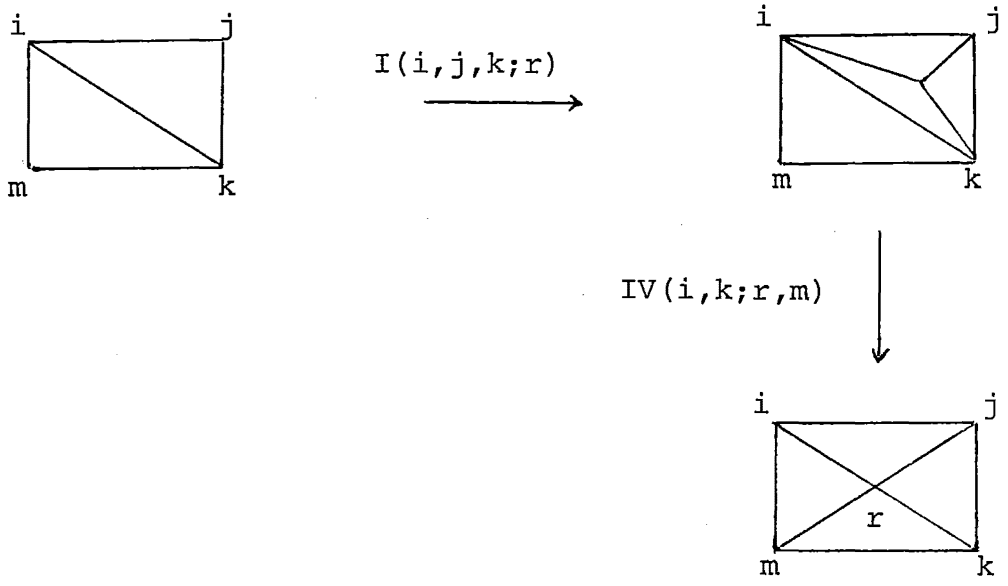
We demonstrate equivalence through construction.

Proof: For notational convenience,

let I(a,b,c;d) indicate the insertion of vertex d in triangle (a,b,c)

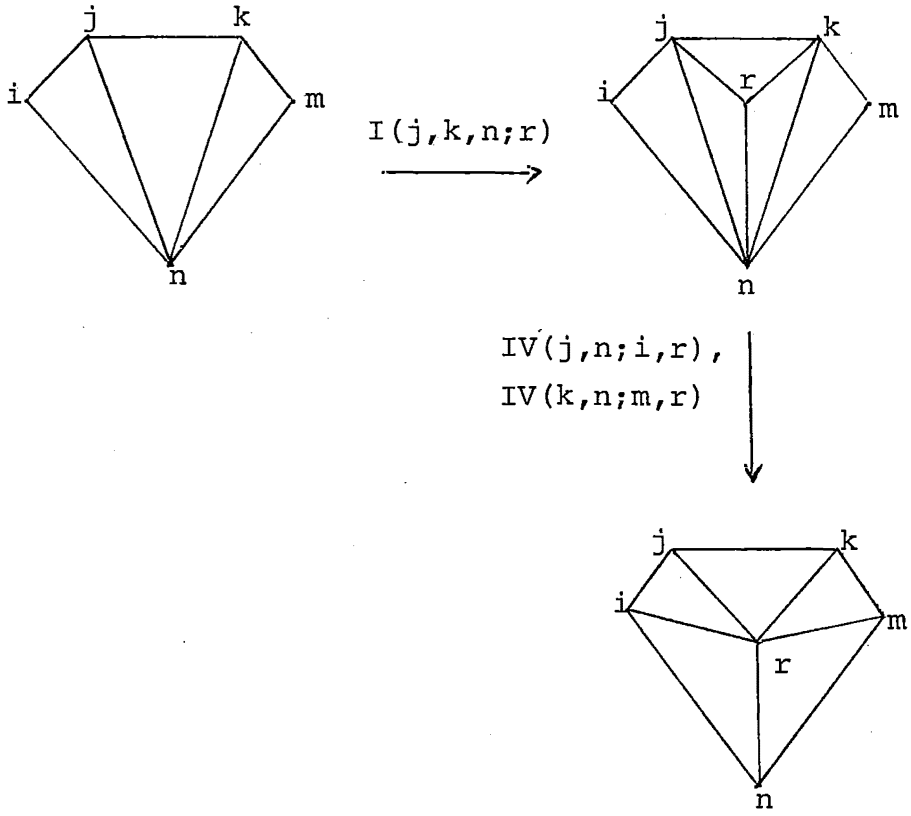
and IV(a,b;c,d) denote the swapping of diagonal (a,b) for diagonal (c,d) in quadrilateral (a,b,c,d).

Then II is equivalent to:




(Alternatively, I(i,k,m;r) followed by IV(i,k;r,j) is equivalent.)

For operation III:




QED

Although we have shown that operations II and III may be decomposed into more elementary operations, we still require an efficient recognition scheme for the structures of Theorem 3.10. We suggest a framework similar to that used in WHEEL_EXPANSION, i.e.: characterise the graph in terms of considering each vertex x as the hub of a unique wheel, $W_x$, and specifying the set of its neighbouring vertices as rim ($W_x$). Using the cyclic nature of the rim ordering, the recognition phase becomes easily attainable, as follows:

Structure of operation I: the triangle:

Rather than using the wheel representation (which would result in duplication), an incremental list, modified during the construction phase as in DELTAHEDRON would be most appropriate.

Structure of operation II: the quadrilateral:

Each edge of a maximal planar graph G = (V,E) is the diagonal of a unique quadrilateral. Consider a vertex $x \in V$ with deg(x) = m > 4, say, and consider the wheel about x, as depicted in Figure 3.11.

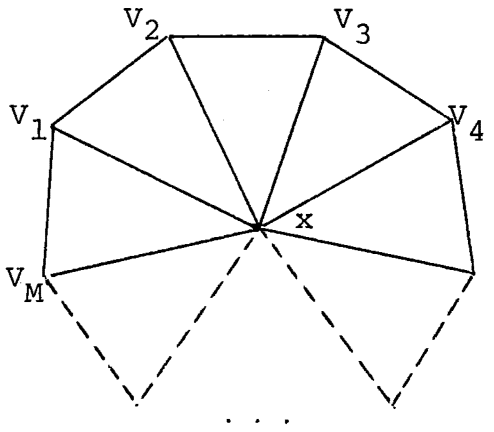



Figure 3.11

Quadrilaterals may be generated in the form

$$[v_i, v_j, v_k, v_r; v_k]$$

where $v_i$ is the hub of the wheel

$v_j, v_k, v_r$ are (cyclically) adjacent vertices of the rim of the wheel

and $(v_i, v_k)$ is the diagonal of the quadrilateral.

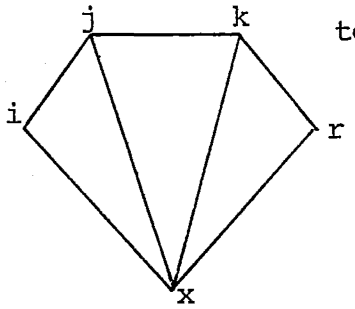
We choose an arbitrary vertex on the rim to initiate the generation. In the example, suppose we start at vertex $v_1$. Then the first quadrilateral is

$$[x, v_m, v_1, v_2; v_1].$$

Incrementing $v_k$ to $v_2$, the next quadrilateral is $[x, v_1, v_2, v_3; v_2]$ and so on, until $[x, v_{m-1}, v_m, v_1; v_m]$.

Applying this technique to all the wheels of G, each quadrilateral would be generated exactly twice. For example, $[x,v_1,v_2,v_3;v_2]$ is equivalent to $[v_2,v_3,x,v_1;x]$. Hence we must be careful to keep track of duplication.

Structure of operation III: the fan:

Define a structure 

to be a fan rooted at x, and denote it by $F(x;i,j,k,r)$. For any $x \in V$, if $deg(x) = 3$ then x clearly cannot be the root of any fan. Hence, x is the root of a fan iff $deg(x) \geq 4$.

Fans may be generated analogously to the quadrilaterals above, via sets of four of the rim vertices of $W_x$ taken in cyclic order. In terms of Figure 3.11, starting again at vertex $v_1$, the first fan is of the form $F(x;v_1,v_2,v_3,v_4)$; this is followed by $F(x;v_2,v_3,v_4,v_5)$ (if $deg(x) \geq 5$), and so on, until $F(x;v_{m-1},v_m,v_1,v_2)$ and finally $F(x;v_m,v_1,v_2,v_3)$.

Remark 3.7 If $deg(x) \geq 4$, then there exist deg(x) fans rooted at x.

Note that there is no duplication in this generation phase, because the root of each fan is unique.

Remark 3.8 Suppose a graph G has vertices $v_1,\ldots,v_n$.

Define $d'(v_i) = \begin{cases} deg(v_i) \text{ if } d(v_i) \geq 4 \\ 0 \text{ otherwise} \end{cases}$

Then G will have $\sum_{i=1}^{n} d'(v_i)$ fans.

Clearly these generation techniques need only be applied locally to the graph G ... i.e.: to the part of G affected by the most recent application of the operations I, II or III. We now outline a straightforward greedy heuristic, WHEEL_GENERATION, based on these operations.

Given a weighted graph (G,w), the application of the described operations yields total weight changes of

I: $w_{ri} + w_{rj} + w_{rk} = T(I;i,j,k,r)$

II: $w_{ri} + w_{rj} + w_{rk} + w_{rm} - w_{ik} = T(II;i,j,k,m,r)$

III: $w_{ri} + w_{rj} + w_{rk} + w_{rn} - w_{jn} - w_{kn} = T(III;i,j,k,m,n,r)$

The WHEEL_GENERATION Heuristic:

1. Initialise by choosing a suitable $K_4$ (INITIAL_2, INITIAL_3)
2. $G = K_4$
3. For each triangle $(i,j,k) \in G$, for each vertex $r \notin V(G)$, calculate $T(I;i,j,k,r)$.
   For each quadrilateral $(i,j,k,m) \in G$, for each vertex $r \notin V(G)$, calculate $T(II;i,j,k,m,r)$.
   For each fan $(i,j,k,m,n) \in G$, for each vertex $r \notin V(G)$, calculate $T(III;i,j,k,m,n,r)$.
   Calculate the maximum increase possible from all these augmentations, and make the corresponding modifications to G.
4. If all vertices have been processed, STOP; otherwise to go step 3.

We also note that operations I, II and III would be used as a subroutine analgously to the wheel expansion operation in algorithm MAXIMAL.

# CHAPTER 4

## MAXIMIZING RELATIONSHIP CHART SCORES OF ALL PAIRS OF FACILITIES

We now consider the first of two extensions to problem ADJACENCY. This chapter investigates the problem of not only optimizing a layout with respect to physical pair-wise adjacencies, but incorporating credit for having pairs of facilities nearly adjacent. Thus we are now seeking a more global solution instead of a myopic local one. As a means of determining a good quality solution to this revised problem, we present modifications of the heuristics DELTAHEDRON and GREEDY of the last chapter. Giffin and Foulds (1983) covers much of the material given here.

### 4.1 The Modified DELTAHEDRON and GREEDY Heuristics

Our definition of adjacency must be extended to allow for near-adjacency. We do this by considering the number of boundaries that must be crossed in travelling from one facility to another. As the actual area desired for a facility is not yet included in our formulation, the distance between boundaries in a layout is assumed to be uniform; without loss of generality we assume a unit distance. The following theorem provides an upper bound on the number of boundaries, $d_n$, which must be traversed between any pair of facilities (including the exterior facility).

Theorem 4.1 [Grünbaum (1967)]

Let $G = (V,E)$ and $n = |V|$. If G is maximal planar then

$$d_n \leq [\frac{n-2}{3}] + 1$$

In order to encapsulate a measure of discrimination between adjacency ratings and near-adjacency ratings, we assume that there exists a sequence of relationship matrices $W_1, W_2, \ldots, W_M$, where $M = [\frac{n-2}{3}] + 1$.

Let $W_k = [w^k_{ij}]_{n\times n}$ for $k = 1,\ldots,M$,

and

$w^k_{ij}$ = the benefit gained by locating facilities i and j, k boundaries apart.

Usually it may be assumed that $w^p_{ij} < w^q_{ij}$ for $p > q$.

Therefore the new model will assume that a building is to be laid out by specifying the number of boundaries separating each pair of facilities. The objective is then to find a layout (or, correspondingly, an adjacency graph) which maximizes the sum of benefits $w^k_{ij}$ taken over all pairs of facilities.

In the context of the adjacency graph, facilities i and j are separated by k boundaries if the shortest path (in terms of the number of edges, under the unit distance assumption) between their corresponding vertices i and j is k. The graph theoretic formulation then becomes:

$$\text{MAXIMIZE} \quad \sum_{\substack{i,j\in V \\ i\neq j}} \sum_{k=1}^{M} w^k_{ij} x^k_{ij}$$

subject to $T = (V,E)$ is maximally planar,

where

$$x_{ij}^{k} = \begin{cases} 1 \text{ if the shortest path between } i,j \text{ has } k \text{ edges} \\ 0 \text{ otherwise} \end{cases}$$

and

$$E = \{i,j : x_{ij}^{1} = 1,\ i,j \in V\}$$

We call this new problem N_BOUNDARY. Note that the case M = 1 corresponds to problem ADJACENCY.

Now we describe how the heuristics DELTAHEDRON and GREEDY may be modified to take into account the new objective function.

An obvious requirement is the need to calculate the shortest path (in terms of the number of edges) between each pair of vertices at each updating step of the adjacency graph construction. This could be achieved using either the Floyd-Roy-Warshall algorithm (Floyd (1962), Roy (1959), Warshall (1962)) or the algorithm of Dantzig (1967). Each of these methods executes in time $0(n^3)$, so that embedding either within the N_BOUNDARY framework would result in complexities of $0(n^5)$ and $0(n^6)$ respectively for the modified versions of DELTAHEDRON and GREEDY.

However, neither of the shortest-path algorithms mentioned takes advantage of the updating nature of the problem; both are "start-over" algorithms in the sense that the information contained in the shortest-path matrix found for a graph G at the $i^{th}$ iteration is not used at the next iteration. We now investigate an adaptation of algorithm DANTZIG that will lead to a more efficient method of determining the shortest paths. Firstly we give a description of algorithm DANTZIG.

Let $D_1, D_2, \ldots, D_n$ be a sequence of matrices, such that $D_k$ is a $k \times k$ matrix containing the shortest path lengths

between vertices $x_1, x_2, \ldots, x_k$,

i.e.: $D_k = [d_k(x_i, x_j)]$, d a distance function.

Let $d_{k+1}(x_i, x_{k+1}) = \underset{1 \le j \le k}{\text{MIN}} [d_k(x_i, x_j) + d_k(x_j, x_{k+1})]$,

$1 \le i \le k$

and

$$d_{k+1}(x_{k+1}, x_i) = \underset{1 \le j \le k}{\text{MIN}} [d_k(x_{k+1}, x_j) + d_k(x_j, x_i)],$$

$1 \le i \le k$

Then

$$d_{k+1}(x_i, x_j) = \text{MIN}[d_k(x_i, x_j), d_{k+1}(x_i, x_{k+1}) + d_{k+1}(x_{k+1}, x_j)],$$

$1 \le i, j \le k$

(These equations define the $(k+1) \times (k+1)$ distance matrix induced by the addition of the $(k+1)$st vertex to the graph.)

Analysis of this rationale suggests an appropriate updating form, as used in Cheston (1976) and Cheston and Corneil (1982).

Suppose we have $D_k$ associated with $G_k = (V_k, E_k)$ and we add vertex $x_{k+1}$ (not in $V_k$), adjacent to $x_1, \ldots, x_r$, say. Then

<u>for</u> $x \in V_k$ <u>do</u>

$d_{k+1}(x_i, x_{k+1}) = \text{MIN}[d_k(x, x_i) + 1]$

$1 \le i \le r$

(since $d_{k+1}(x_i, x_{k+1}) = 1$)

and

<u>for</u> $u \in V_k$

<u>for</u> $x \in V_{k+1} - u$ <u>do</u>

$d_{k+1}(u, x) = \text{MIN}[d_k(u, x), d_{k+1}(u, x_{k+1}) + d_{k+1}(x, x_{k+1})]$

This updating form has time complexity $O(n^2)$ and is applicable to general graphs. In dealing with maximal planar graphs, however, further simplifications are possible, which we now discuss.

For the DELTAHEDRON heuristic, suppose that we have the implicit vertex renumbering from INITIAL_3, such that at each iteration k we insert vertex $x_k$ into triangle $T(x_r,x_s,x_t)$, say. Then the new shortest-path matrix, $D_{k+1}$, may be found from the current $D_k$ as follows:

(1) $d_{k+1}(x_{k+1},x_r) = d_{k+1}(x_{k+1},x_s) = d_{k+1}(x_{k+1},x_t) = 1$

(2) $d_{k+1}(x_{k+1},x_i) = MIN[d_k(x_r,x_i),d_k(x_s,x_i),d_k(x_t,x_i)] + 1,$
$i \neq r,s,t,k+1$

(since the shortest path from $x_i$ to $x_{k+1}$ must pass through <u>one</u> of $x_r$, $x_s$, $x_t$)

(3) $d_{k+1}(x_i,x_j) = d_k(x_i,x_j)$, $i,j \neq k+1$
(since the addition of vertex $x_{k+1}$ does not change the <u>length</u> of any existing shortest path)

(4) create the corresponding symmetric matrix entries for cases (1) and (2).

Thus, $D_{k+1}$ may be obtained by augmenting $D_k$ by one row (and its transpose). We will call an implementation of DELTAHEDRON containing these procedures N_BOUNDARY_DELTAHEDRON.

<u>Remark 4.1</u> N_BOUNDARY_DELTAHEDRON has time complexity $O(n^3)$.

Now we consider similar modifications to GREEDY. Suppose $G_k = (V_k,E_k)$ is the graph constructed by GREEDY after the addition of the $k^{th}$ edge ($k < 3n-6$), and suppose that the next candidate edge is $(x_i,x_j)$. Then there are three

possibilities:

(1) $x_i, x_j \in V_k$.

We use the following edge-addition algorithm (due to Cheston (1976)).

For any two vertices p and q, the only possible way that the addition of edge $(x_i, x_j)$ can lead to a shorter path between p and q is if the new path is via $p - x_i - x_j - q$ or $p - x_j - x_i - q$. This gives an $o(n^2)$ update algorithm:

for $(p,q) \in V \times V$ do

$$d_{k+1}(p,q) = \text{MIN}[d_k(p,q), d_k(p,x_i) + 1 + d_k(x_j,q), d_k(p,x_j) + 1 + d_k(x_i,q)]$$

(since $d_{k+1}(x_i,x_j) = d_{k+1}(x_j,x_i) = 1$).

(2) $x_i \in V_k$ or $x_j \in V_k$.

Suppose, without loss of generality, that $x_i \in V_k$ and $x_j \notin V_k$. Then

$$d_{k+1}(x_m,x_j) = d_k(x_m,x_i) + 1, \quad m \neq i,j$$

and $d_{k+1}(x_i,x_j) = d_k(x_j,x_i) = 1$

(3) $x_i, x_j \notin V_k$

(a) $d_{k+1}(x_i,x_j) = d_{k+1}(x_j,x_i) = 1$

(b) $d_{k+1}(x_m,x_i) = L \quad , \quad m \neq i,j$

(c) $d_{k+1}(x_m,x_j) = L \quad , \quad m \neq i,j,$

where L is a sufficiently large number.

The implementation of these simplifications results in a method we will label N_BOUNDARY_GREEDY, with a worst-case complexity that has been reduced only to $0(n^5)$.

The structure of this heuristic is not completely analogous to GREEDY itself; at each augmentation step, the next edge to be added is that which leads to the greatest

increase in the N_BOUNDARY objective function. This implies that each acceptable edge not part of the current solution is examined at each iteration, rather than using the GREEDY weighted ordering. (Acceptability here is in terms of planarity violation.)

An investigation of the behaviour of N_BOUNDARY_GREEDY, shows that an approximation with a much simpler characterisation may be developed, whose results differ little from the original. In order to maximize the n-boundary benefit, N_BOUNDARY_GREEDY seeks to construct a shortest-path matrix whose maximum element is 2. It accomplishes this by creating a vertex of degree n-1 at the earliest opportunity; intuitively, the reason for this is as follows. Assume that $w_{ij}^k > w_{ij}^r \Leftrightarrow k < r$, and suppose that the most highly weighted edge is $(\alpha,\beta)$. Then there exist three choices for the next edge addition:

(i) edge $(x,\alpha)$, with incremental value $w_{x\alpha}^1 + w_{x\beta}^2$

(ii) edge $(x,\beta)$, with incremental value $w_{x\beta}^1 + w_{x\alpha}^2$

(iii) edge $(x,y)$, with incremental value $w_{xy}^1$

Hence, generally the best choice will be either of (i) or (ii), except in a pathological case, where, for example $w_{\alpha\beta} >> w_{\alpha i}, w_{\beta i}$ $\forall$ i. Continuing in this way, it turns out that the most common sequence of additions creates either $\alpha$ or $\beta$ as the hub of a wheel, whose rim is gradually built up after most vertices (of $V - \{\alpha,\beta\}$) have been added. We call such a structure an umbrella. The approximation method N_BOUNDARY_GREEDY_APPROX, given in Figure 4.1 attempts to mimic the behaviour of N_BOUNDARY_GREEDY; small differences (or up to 1%) in final solution value occur, especially for

higher values of n, because, as the degree of the hub vertex $\gamma$, say, approaches n-1, edges other than [ $(x,\gamma)$ : $x \notin v$ (umbrella)] may be preferred additions. N_BOUNDARY_GREEDY uniformly dominates N_BOUNDARY_GREEDY_APPROX. The weight-based addition order of GREEDY is also retained, but appears to sacrifice little in terms of solution quality. Table 4.1 indicates the average solution times required for the two heuristics over a series of test problems; N_BOUNDARY_DELTA-HEDRON also exhibits the tendency towards producing an umbrella structure. From the implementation viewpoint this is undesirable: constructing a practical layout corresponding to such a structure would be difficult (and maybe even impossible, if the area of the hub facility were small). A strategy for overcoming this problem will be discussed in section 4.3

| N | CPU-time (Burroughs B6930 secs) | |
|---|---|---|
| | N_BOUNDARY_GREEDY | N_BOUNDARY_GREEDY_APPROX |
| 20 | 374.20 | 12.42 |
| 30 | 2890.19 | 70.29 |
| 40 | 12113.76 | 167.41 |

Table 4.1 Average solution times for N_BOUNDARY_GREEDY and N_BOUNDARY_GREEDY_APPROX

In both the greedy procedures for problem N_BOUNDARY, the planarity testing phase use the algorithm of Hopcroft and Tarjan (1974). Simplification strategies proved fruitful in reducing processing times for the shortest-path calculations in algorithm DANTZIG, so we now pursue a similar methodology in attempting to overcome the computational burden that the Hopcroft and Tarjan test represents.

```
Procedure N_BOUNDARY_GREEDY_APPROX;
begin
    ordered_list(*) := [ (x_i,x_j) ordered according to non-
          increasing weight w(x_i,x_j)];
    (x_k,x_r) := ordered_list(1);
    select x_a such that w(x_k,x_a) = MAX   w(x_k,x_m);
                                     x_m ≠ x_r
    select x_b such that w(x_b,x_r) = MAX   w(x_m,x_r);
                                     x_m ≠ x_k
    if w(x_k,x_a) ≥ w(x_r,x_b)
    then hub := x_k else hub := x_r;
    umbrella := [ (hub,x_m):x_m ≠ hub];
    (* V(umbrella) = V(G) *)
    count := 0;
    i := 2;
    while count < 2n-5 do
    begin
        if ordered_list(i) ∉ umbrella
        then
            if umbrella + ordered_list(i) PLANAR
            then begin
                umbrella := umbrella + ordered_list(i);
                count := count + 1;
            end;
            i := i + 1;
    end
end;
```

Figure 4.1 Procedure N_BOUNDARY_GREEDY_APPROX

The first amendment is a slight modification of the greedy concept: we now require that the final construction must contain a Hamiltonian circuit (cycle). The following theorem shows that we are prejudicing the solution only slightly:

Theorem 4.2 [Whitney (1931)]
Let G be a maximal planar graph with no separating triangle. Then G has a Hamiltonian circuit.

We fulfill the requirement at the initialisation stage by constructing a Hamiltonian circuit, HC, in a greedy fashion according to procedure MAX_HAM (Korte (1979), Fisher, Nemhauser and Wolsey (1979)). A pidgin-PASCAL version of MAX_HAM is given in Figure 4.2.

```
Procedure MAX_HAM;
begin
    ordered_list(*) := [e_i ∈ E(G) ordered according to non-
          increasing weight w_ij]
    HC := ordered_list(1);
    i = 1;
    repeat
        i = i + 1;
        if HC + ordered_list(i) creates a subcircuit
        then reject ordered_list(i)
        else HC := HC + ordered_list(i)
    until i = n-1;
    complete HC by joining the two unattached vertices;
    output HC;
end;
```

Figure 4.2 Procedure MAX_HAM

This method guarantees the creation of a circuit with weight at least half that of the optimal weight. To guarantee a weight ratio of at least $\frac{2}{3}$, the alternative procedure MATCH_HAM (Fisher, Nemhauser and Wolsey (1979)), as described in Figure 4.3, may be used. In the results presented, only MAX_HAM was invoked.

```
Procedure MATCH_HAM;
begin
    find a maximum weight perfect 2-matching M in G;
    if M is not a circuit
    then begin
        for each circuit Z_i in M do
        begin
            delete the smallest weight edge from Z_i;
        end;
        complete HC by replacing the deleted edges by any
              subset of edges yielding a circuit;
    end;
    output HC;
end;
```

Figure 4.3 Procedure MATCH_HAM

Let emb(HC) be a planar embedding of HC, and let G* be the graph we are constructing. Since V(HC) = V(G*), all further edges added to G* will now correspond to edges joining elements of HC, and G will remain planar if every edge of E(G*) - E(HC) can be drawn (in the graph theoretic sense) in either face of the polygon formed by emb(HC). That is, G will be planar if all the edges of E(G*) - E(HC) can be partitioned into two sets, one set embedded on the "inside" of

emb(HC) and the other set on the "outside" without crossings. (We will return to this planarity characterisation in more detail in the next chapter.)

Definition 4.1 The auxiliary graph, A, of G* is defined by

$$a_e \in V(A) \text{ iff } e \in E(G^*) - E(HC)$$

and

$$e = (e_1, e_2) \in E(A) \text{ iff } e_1, e_2 \text{ cross in } emb(G^*)$$

The following theorem summarizes the above notions.

Theorem 4.3 G is planar iff A is bipartite.

Proof: [See, for example, Saaty and Kainen (1976)].

Updating planarity testing with the Hamiltonian circuit requirement takes the form of GREEDY_UPDATE, as shown in Figure 4.4. The key to efficiency in the procedure is the check for bipartiteness: does A remain bipartite after the addition of an edge $e_i$? (If so, we accept $e_i$ as part of the GREEDY_UPDATE solution, G*, and update G* and A permanently; otherwise we reject $e_i$ and restore G* and A to their previous states.) Essentially we require the rapid detection of an odd (length) circuit in A, as indicated by an inconsistent assignment of colours to vertices.

Let comp_no be the number of components in A and comp(x) denote the component identifier of vertex x in A.

Suppose edges i and j cross in emb(G*). Then

```
Procedure GREEDY_UPDATE;
begin
    (* given : a candidate edge e ∉ E(G*) *)
    V(A) := V(A) + a_e;
    accepted := true;
    no_more_crossings := false;
    repeat
        determine an edge f which e crosses in emb(G*);
        (* relative to some cyclic ordering of V(HC) *)
        if f = φ then no_more_crossings := true
        else begin
            E(A) := E(A) + (e,f);
            if A not bipartite
            then begin
                accepted := false;
                V(A) := V(A) - a_e;
                delete all edges (e,*) ∈ E(A);
            end;
        end;
    until (no_more_crossings or (not accepted));
end;
```

Figure 4.4 Procedure GREEDY_UPDATE

```
(i)   if a_i,a_j ∉ V(A)
      then begin
          colour(a_i) := 0;
          colour(a_j) := 0;
          comp_no := comp_no + 1;
          comp(a_i) := comp_no;
          comp(a_j) := comp_no;
      end;
(ii)  if a_i ∈ V(A), a_j ∉ V(A)
      then begin
          colour(a_j) := (colour(a_i) + 1) mod 2;
          comp(a_j) := comp(a_i);
      end;
      if a_i ∉ V(A), a_j ∈ V(A)
      then begin
          colour(a_i) := (colour(a_j) + 1) mod 2;
          comp(a_i) := comp(a_j);
      end;
(iii) if a_i,a_j ∈ V(A) then
          if comp(a_i) = comp(a_j) then
              if colour(a_i) = colour(a_j) then
                  G NON-PLANAR (* odd circuit found
                                  in A*)
              else (* no changes required *)
          else begin (* comp(a_i) ≠ comp(a_j) *)
              for each a_k ∈ comp(a_j) do
                  comp(a_k) := comp(a_i);
              if colour(a_i) = colour(a_j) then
                  for each a_k ∈ comp(a_j) do
```

colour($a_k$) := (colour($a_k$) + 1) mod 2;

end;

Note that two colours (0 and 1) are sufficient. Isolated vertices are arbitrarily assigned colour 0; independent components are built up using this arbitrary colour as the root. If an edge between two components is added and the colours of the end vertices coincide, all that is required is that the colours of the vertices of one of the components be reversed (i.e.: 0 <-> 1) and the components then amalgamated. As edge crossings in G* caused by the addition of a new edge are detected, the corresponding implied adjacency is immediately checked in A for the violation of bipartiteness. If any violation occurs the edge is rejected, saving the need for determining further crossings unnecessarily.

Figure 4.5 contains an expanded version of Figure 4.4, incorporating the ideas mentioned above. Table 4.2 indicates the comparative performance of GREEDY_UPDATE relative to that of GREEDY (again using problems with edge weights generated using the Box-Muller method).

```
Procedure GREEDY_HAMILTONIAN;
Procedure UPDATE_A;
begin
    V(A) := V(A) + a_{s,f};
    if number_of_crossings = 0
    then begin
        define comp(a_{s,f});
        colour(a_{s,f}) := 0; (* wlog *)
    else begin
        E(A) := E(A) + stored candidate edges;
```

```
        colour(a_{s,f}) := (colour(adj_comp(1)) + 1) mod 2;
        for each adj_comp(i), i > 1, do
        begin (* establish inter-component consistency in A *)
            if comp_col(adj_comp(i)) <> colour(a_{s,f})
            then (* flip-flop *)
            for each v ∈ adj_comp(i) do
                colour(v) := (colour(v) + 1) mod 2;
        end;
    end;
    reinitialise all temporary storage;
    update adj(), aux();
end;
Function check_A_bipartite : boolean;
begin (* newly-formed adjacency between a_{next,v}, a_{s,f} *)
    c := comp(a_{next,v});
    col := colour(a_{next,v});
    if first(c) (* initial induced edge to c in A *)
    then begin
        check_A_bipartite := true;
        first(c) := false;
        comp_col(c) := col;
    end
    else
    if col = comp_col(c)
    then check_A_bipartite := true;
    else check_A_bipartite := false;
end;
Function PLANAR(G + e_k) : boolean;
begin
```

```
    (* suppose e_k represents edge (x,y) *)
    distxy := |φ(x) - φ(y)|;
    distyx := |φ(y) - φ(x)|;
    (* minimize checking for crossings *)
    if distxy <= distyx
    then begin
        s := x; f := y;
    end
    else begin
        s := y; f := x;
    end;
    next := adj_HC(s); (* adjacent vertex in HC in sense of
          increasing φ *)
    accepted := true;
    repeat
        for v ∈ adj(next) do
        (* adj() contains adjacencies to next in G *)
        if edges (next,v) and (s,f) cross
        then
        if check_A_bipartite
        then begin
            no_of_crossings := no_of_crossings + 1;
            temporarily store implied A adjacency in aux();
        end;
        else accepted := false;
        next := adj_HC(next);
    until ((next = f) or (not accepted);
    if accepted then PLANAR := true else PLANAR := false;
end;
```

```
begin (* mainline for GREEDY_HAMILTONIAN *)
    generate maximum weight Hamiltonian circuit, HC, using
          procedure MAX_HAM;
    create HC adjacency matrix adj_HC;
    for each e_i ∈ HC do acceptable(e_i) := false;
    (* retain ordered_list() *)
    create homomorphism φ such that
          φ(v) = k if v is vertex k(mod n) of HC,
           traversed in order;
    G* := HC;
    i := 1; k := 1;
    while i ≤ 2n-6 do
    begin
        identify e_k = ordered_list(e_k)
        if acceptable(e_k)
        then (* viable candidate *)
        if PLANAR(G* + e_k)
        then begin
            G* := G* + e_k;
            W(G*) := W(G*) + w(e_k);
            i := i + 1;
            UPDATE_A;
        end;
    k := k + 1;
    end;
    output G*, W(G*);
end;
```

Figure 4.5 Expanded version of GREEDY_UPDATE

Table 4.2 Comparison of GREEDY and GREEDY_UPDATE Performance

| N | σ | GREEDY | | GREEDY_UPDATE | |
|---|---|---|---|---|---|
| | | Σ* | T* | Σ | T |
| 10 | 5 | 2468 | 1.9 | 2468 | 0.07 |
| 10 | 15 | 2612 | 2.0 | 2606 | 0.06 |
| 10 | 25 | 2783 | 2.0 | 2754 | 0.07 |
| 20 | 5 | 5622 | 20.8 | 5628 | 0.32 |
| 20 | 15 | 6147 | 20.5 | 6076 | 0.27 |
| 20 | 25 | 6819 | 19.9 | 6770 | 0.27 |
| 30 | 5 | 8833 | 69.8 | 8827 | 0.81 |
| 30 | 15 | 9766 | 71.4 | 9694 | 0.73 |
| 30 | 25 | 10765 | 70.5 | 10511 | 0.62 |
| 40 | 5 | 12102 | 165.4 | 12072 | 1.72 |
| 40 | 15 | 13490 | 174.6 | 13387 | 1.50 |
| 40 | 25 | 14777 | 158.9 | 14652 | 1.61 |

*Σ represents solution score, T represents CPU-time.

Solution quality achieved by GREEDY_UPDATE is reduced only slightly compared to that of GREEDY, but significant savings in processing time are available. This is in spite of the fact that the time complexity of GREEDY_UPDATE remains $0(n^3)$, as for GREEDY; only the overhead has been improved. Using MATCH_HAM instead of MAX_HAM as the initialisation procedure may well produce solutions on a par with those of GREEDY, with CPU-times that challenge even DELTAHEDRON.

Having developed the methodology for improving the efficiency of GREEDY, we can now embed GREEDY_UPDATE within

the n-boundary framework. The straightforward implementation N_BOUNDARY_GREEDY_UPDATE, is given in Figure 4.6.

```
Procedure N_BOUNDARY_GREEDY_UPDATE;
begin
    ordered_list(*) := [ edges e_i ordered according to non-
          increasing weight w]
    G* := (V(G),φ);
    find a maximum weight Hamiltonian circuit
          HC = V(G*),E(HC)) in G;
    tot_ben := Σ_{e∈E(HC)} w(e);
    ordered_list(*) := ordered_list(*) - {e : e ∈ E(HC)};
    SP(*) := shortest-path matrix induced by HC;
    count := n;
    max_ben := -1;
    k := minimum index in ordered_list;
    while count < 3n-6 do
    begin
        if G* + ordered_list(k) planar
            (* using GREEDY_UPDATE *)
        then begin
            ben_inc := benefit induced by addition of
                  ordered_list(k);
            if ben_inc > max_ben
            then begin
                max_ben := ben_inc;
                max_pt := k;
            end;
        end
```

```
        else ordered_list := ordered_list-ordered_list(k);
        if k < 3n-6 then k := k + 1
        else begin
            G* := G* + ordered_list(max_pt);
            update SP(*);
            tot_ben := tot_ben + max_ben;
            ordered_list := ordered_list-ordered_list(max_pt);
            k := minimum index in ordered_list;
        end;
    end;
    output G*, tot_ben;
end;
```

Figure 4.6 N_BOUNDARY_GREEDY_UPDATE

## 4.2 Computational Experience with the n-boundary heuristics

Firstly we compare N_BOUNDARY_DELTAHEDRON and N_BOUNDARY_GREEDY. Both methods produce solutions of similar quality, but the performance of N_BOUNDARY_GREEDY_UPDATE remains disappointing in relation to that of N_BOUNDARY_GREEDY in terms of CPU-time. However, problems with n = 50 are now tractable. Analysis of Procedure call statistics made during program execution shows that the shortest-path calculations required to determine the next best candidate edge at each iteration have by far the most dominant time factor. For example, in a problem with n = 40, fully ninety percent of the total CPU-time is devoted to this task, whereas each planarity test is executed on average in 1200 μ sec. Table 4.3 gives a sample of the results obtained.

Little further improvement therefore seems likely using the greedy rationale, given that the updating version of the shortest-path calculations is included; for N_BOUNDARY_GREEDY these simplifications are nowhere near as efficient as those for N_BOUNDARY_DELTAHEDRON.

We noted earlier the observed tendency for both N_BOUNDARY_DELTAHEDRON and N_BOUNDARY_GREEDY to produce at least one vertex of high degree in the adjacency graph - the so-called "umbrella" effect. Naturally, this also occurs for N_BOUNDARY_GREEDY_UPDATE. Given that such an occurrence

Table 4.3: N_BOUNDARY_DELTAHEDRON vs N_BOUNDARY_GREEDY

| N | σ | N_BOUNDARY_DELTAHEDRON | | N_BOUNDARY_GREEDY | |
|---|---|---|---|---|---|
| | | Σ | T | Σ | T |
| 10 | 5 | 3620 | 0.22 | 3613 | 3.15 |
| 10 | 15 | 3581 | 0.23 | 3581 | 2.85 |
| 10 | 25 | 3671 | 0.24 | 3653 | 2.87 |
| 20 | 5 | 17000 | 1.21 | 17013 | 119.49 |
| 20 | 15 | 17077 | 1.22 | 17057 | 102.26 |
| 20 | 25 | 17072 | 1.22 | 17069 | 96.36 |
| 30 | 5 | 40023 | 3.91 | 40025 | 888.78 |
| 30 | 15 | 40205 | 3.89 | 40217 | 741.75 |
| 30 | 25 | 40433 | 3.94 | 40287 | 851.22 |
| 40 | 5 | 73086 | 9.17 | 73082 | 3441.98 |
| 40 | 25 | 73179 | 9.19 | 73173 | 2319.58 |
| 50 | 5 | 116539 | 17.89 | 116533 | 11046.24 |

may lead to difficulties in translating from the adjacency graph to a corresponding practical layout, we attempt to alleviate the problem by simply constraining the maximum degree of any vertex to some prescribed limit. For any vertex, v say, we denote this limit DEG_CONST(v). In applying this restriction to problem N_BOUNDARY, the DEG_CONST values chosen are essentially arbitrary; in Chapter 6 they are made a function of facility area. Unfortunately, it may not be possible to construct a maximal planar graph for a given set of DEG_CONST values (for similar reasons to general graph theory problems of non-realizability of degree sequences).

This situation shows up in the execution of N_BOUNDARY_DELTAHEDRON in the form of being unable to find an insertion triangle whose vertex degrees are all at levels below their DEG_CONST limits. In N_BOUNDARY_GREEDY_UPDATE the symptom is having no valid candidate edge left to add, yet the current adjacency graph is not maximally planar. A safeguard must therefore be introduced to eliminate such occurrences. For N_BOUNDARY_DELTAHEDRON this involves a perturbation of the DEG_CONST value of one or more vertices whose degree at the current stage of the construction has attained its limit as a results of the latest insertion. For N_BOUNDARY_GREEDY_UPDATE we must keep a record of which vertex (or vertices) first reaches its (their) degree limit so that these may be incremented first; the construction must essentially restart from this initial level so as not to prejudice the solution with the addition of further edges chosen only because of the tight constraints (whereas a slight relaxation of the constraints may permit a not only significantly superior, but

also a completable, result). In practice the perturbation is required infrequently in N_BOUNDARY_DELTAHEDRON, but surprisingly frequently in N_BOUNDARY_GREEDY_UPDATE, resulting in increases in CPU-time as the heuristic iterations through the perturbations endeavouring to find a feasible solution. This is offset, however, by the reduced number of alternatives available to explore, resulting in an overall reduction in processing time.

Tables 4.4 to 4.6 give a sample of the performance results for the two methods. Direct comparison of solution quality is possible only with the use of equal values of DEG_CONST in each method (this value is initially applied uniformly to all vertices). In all observed cases, the DEG_CONST value for which a feasible solution was attainable using N_BOUNDARY_DELTAHEDRON was (often substantially) smaller than for N_BOUNDARY_GREEDY_UPDATE. Table 4.6 shows that the degree constraint imposition leads to a marked increase in the diameter (the length of the longest shortest path) in the adjacency graph.

As expected, the solution values for the constrained problems are generally lower than for the corresponding unconstrained cases; two exceptions are noted in Table 4.5. For equivalent values of DEG_CONST, N_BOUNDARY_GREEDY_UPDATE gives superior performance on smaller problems, at the expense of the considerable CPU-time requirement. By n = 40, however, N_BOUNDARY_DELTAHEDRON dominates, while still requiring approxmately one-five hundredth of the processing resources.

Comparing Tables 4.1 and 4.4, it may be noted that N_BOUNDARY_GREEDY_UPDATE is much slower than N_BOUNDARY_GREEDY_APPROX. This is to be expected because of the

Table 4.4: Comparison of N_BOUNDARY_DELTAHEDRON and N_BOUNDARY_GREEDY_UPDATE solution values for constrained problems

| N | σ | DEG_CONST* | N_BOUNDARY_DELTAHEDRON Σ | N_BOUNDARY_DELTAHEDRON T | N_BOUNDARY_GREEDY_UPDATE Σ | N_BOUNDARY_GREEDY_UPDATE T |
|---|---|---|---|---|---|---|
| 10 | 5 | 7 | 3526 | 0.21 | 3613 | 3.22 |
| 10 | 10 | 7 | 3698 | 0.20 | 3737 | 3.17 |
| 10 | 15 | 7 | 3569 | 0.22 | 3581 | 3.02 |
| 10 | 20 | 7 | 3774 | 0.22 | 3807 | 3.05 |
| 10 | 25 | 7 | 3525 | 0.20 | 3605 | 2.96 |
| 20 | 5 | 12 | 16670 | 0.86 | 16691 | 104.98 |
| 20 | 10 | 8 | 15747 | 0.65 | 16407 | 81.90 |
| 20 | 15 | 10 | 16398 | 0.69 | 16560 | 82.16 |
| 20 | 20 | 8 | 15855 | 0.58 | 16445 | 76.99 |
| 20 | 25 | 9 | 16515 | 0.66 | 16625 | 82.55 |
| 30 | 5 | 11 | 37887 | 1.53 | 38246 | 525.14 |
| 30 | 10 | 10 | 37440 | 1.29 | 38192 | 480.82 |
| 30 | 15 | 12 | 38446 | 1.63 | 38680 | 505.96 |
| 30 | 20 | 17 | 39675 | 2.27 | 39331 | 572.84 |
| 30 | 25 | 13 | 38581 | 1.59 | 38887 | 577.04 |
| 40 | 5 | 20 | 71524 | 5.37 | 70988 | 2353.91 |
| 40 | 10 | 15 | 70313 | 3.34 | 69939 | 2027.25 |
| 40 | 25 | 16 | 71034 | 4.67 | 70535 | 2090.35 |

* This value represents the smallest value of DEG_CONST for which a feasible solution was found for N_BOUNDARY_GREEDY_UPDATE.

Table 4.5: N_BOUNDARY_DELTAHEDRON solution values at the minimal achievable value of DEG_CONST*

| N** | σ | DEG_CONST | Σ | T |
|---|---|---|---|---|
| 20 | 5 | 7 | 16077 | 0.48 |
| 20 | 10 | 7 | 16069*** | 0.46 |
| 20 | 15 | 7 | 15679 | 0.46 |
| 20 | 20 | 7 | 15992*** | 0.47 |
| 20 | 25 | 7 | 16265 | 0.47 |
| 30 | 5 | 7 | 37603 | 0.83 |
| 30 | 10 | 7 | 35946 | 1.08 |
| 30 | 15 | 8 | 35785 | 1.06 |
| 30 | 20 | 8 | 36912 | 1.02 |
| 30 | 25 | 8 | 36379 | 0.97 |
| 40 | 5 | 8 | 64804 | 1.79 |
| 40 | 10 | 8 | 64846 | 1.60 |
| 40 | 25 | 8 | 63950 | 1.65 |

* values for the same problems as in Table 4.4

** for n = 10 see Table 4.4

*** note that these values are actually higher than their corresponding values in Table 4.4

Table 4.6: The effect of degree constraint on maximal planar graph diameter*; n = 10, 20, 30.

| N | σ | N_BOUNDARY_DELTAHEDRON | | N_BOUNDARY_GREEDY_UPDATE | |
|---|---|---|---|---|---|
| | | UNCONSTRAINED | CONSTRAINED | UNCONSTRAINED | CONSTRAINED |
| 10 | 5 | 2 | 3 | 2 | 2 |
| 10 | 10 | 2 | 3 | 2 | 2 |
| 10 | 15 | 2 | 2 | 2 | 2 |
| 10 | 20 | 2 | 3 | 2 | 2 |
| 10 | 25 | 2 | 3 | 2 | 2 |
| 20 | 5 | 2 | 3 | 2 | 3 |
| 20 | 10 | 2 | 5 | 2 | 3 |
| 20 | 15 | 2 | 4 | 2 | 3 |
| 20 | 20 | 2 | 4 | 3 | 4 |
| 20 | 25 | 2 | 4 | 3 | 4 |
| 30 | 5 | 2 | 5 | 2 | 4 |
| 30 | 10 | 2 | 4 | 2 | 4 |
| 30 | 15 | 2 | 4 | 2 | 4 |
| 30 | 20 | 2 | 3 | 2 | 3 |
| 30 | 25 | 2 | 4 | 3 | 4 |

* This represents the length of the longest n-boundary path in the solution

simplification used in the latter. N_BOUNDARY_GREEDY_APPROX was designed for the unconstrained N_BOUNDARY_PROBLEM, not for the degree restriction case. While it would be possible to modify the routine to take this into account, the idea was not pursued because solution quality was deemed the important factor. Also processing time requirements would necessarily increase with the added restrictions, since the number of edges in the 'umbrella' could not exceed DEG_CONST. For multi-scenario investigations requiring repeated application of the heuristic, however, degree-constrained N_BOUNDARY_GREEDY_APPROX could be a useful alternative.

Overall computational experience indicates that N_BOUNDARY_DELTAHEDRON and N_BOUNDARY_GREEDY_UPDATE represent successful extensions to DELTAHEDRON and GREEDY, providing good quality solutions to the N_BOUNDARY problem. N_BOUNDARY_DELTAHEDRON uniformly outperforms N_BOUNDARY_GREEDY_UPDATE in terms of CPU-time, due to their complexity differences. While at equal degree constraint levels N_BOUNDARY_GREEDY_UPDATE often produced solutions of higher value, often the amount of constraint violation over the input level necessary to obtain feasibility reduced the value of the solution significantly when the difficulty of its implementation as a plan is considered. Only N_BOUNDARY_DELTAHEDRON could generally provide practical and efficient solutions of high quality.

The main drawback of the N_BOUNDARY formulation is the definition of the sequence of relationship matrices $W_k$. Unless these may be derived for $k > 1$ from the adjacency relationship matrix, $W_1$, by some scaling process based on a distance measure, discriminatory estimates of values may be

difficult to obtain. An alternative and more attractive extension of problem ADJACENCY which incorporates facility areas and a more intuitive distance measure is investigated in Chapter 6.

# CHAPTER 5

## GRAPH PLANARITY TESTING IN AN UPDATING ENVIRONMENT

In Chapter 4 we noted that the naive form of the GREEDY heuristic took no advantage of the updating nature of the problem, namely that at each stage we are considering the addition of only one edge, e, to an already planar graph, G. Often this means that the structural modifications to the graph resulting from this change will be local, so that a planarity test of the full graph G + e is not required. In this chapter we discuss some methods of improving the efficiency of the greedy approach; this material also appears in Giffin and Foulds (1984).

Christofides, Galliani and Stefanini (1980) suggest utilising some properties of planar graphs as an initial improvement step, as follows:

Let a planar graph G be composed of components $C_1, C_2, \ldots, C_k$, where each component consists of bicomponents $B_{p1}, \ldots, B_{pk_p}$ (a bicomponent is a 2-vertex connected subgraph).

Theorem 5.1 [Whitney (1932)]
G is planar iff each of its bicomponents is planar.

Property 5.1 If $x_i \in C_{p_1}$, $x_j \in C_{p_2}$ $(p_1 \neq p_2)$, then $G + (x_i, x_j)$ is planar.

Property 5.2 If $x_i, x_j \in B_{pq}$, then $G + (x_i, x_j)$ is planar iff $B_{pq} + (x_i, x_j)$ is planar.

Property 5.3 If $x_i \in B_{pq_1}$, $x_j \in B_{pq_2}$ $(q_1 \neq q_2)$, then $G + (x_i,x_j)$ is planar iff the bicomponent of $C_p + (x_i,x_j)$ containing $(x_i,x_j)$ is planar.

These results could be incorporated into the Hopcroft and Tarjan planarity testing procedure in its initial phase, which splits the graph G into its bicomponents in accordance with Theorem 5.1. However, the updating environment is one for which the Hopcroft and Tarjan algorithm is not entirely suited: it is the depth-first-search (DFS) tree representation of a graph that allows a linear-time complexity for the procedure, but such a characterisation is not immediately available in the present problem. Here the data structure is dynamic, in the sense that a complete transversal of the graph (which would then admit an efficient representation) is possible only after the final edge addition. Until that stage the DFS description must be constantly reconstructed, as many forms of edge addition could destroy its validity. Similar comments may also be applied to the vertex-addition planarity testing algorithm of Lempel, Even and Cederbaum (1967) and the so-called "Left-Right" algorithm of de Fraysseix and Rosenstiehl (1981).

Before presenting our improvement approach, we now briefly outline the classical approach to testing the planarity of a given finite graph (see, for example, Even (1979)).

Let $G = (V,E)$ be a non-separable graph, and $V_C \subset V$. Consider a partition of $V - V_C$ into <u>classes</u>, such that two vertices, $x_1,x_2$, are in the same class if and only if there is a path connecting them which does not pass through any vertex of $V_C$. Each class K defines a subgraph $B_i = (V_i,E_i)$,

where $V_i$ consists of the vertices of K plus the vertices of $V_C$ which are connected by an edge to a vertex of K, and $E_i$ contains all edges of G which have at least one end vertex in K. Such components are called bridges, and vertices of a component which are members of $V_C$ are called attachments.

Let C be a simple circuit of a non-separable graph G, and $B_1,\ldots,B_k$ be the bridges with respect to C. We say that $B_i$ and $B_j$ interlace if at least one of the following conditions holds:

(i) there are two attachments of $B_i$, a and b say, and two attachments of $B_j$, c and d say, such that all four are distinct and they appear on C in the order a,c,b,d

(ii) there are three attachments common to $B_i$ and $B_j$.

Theorem 5.2 Let G be a non-separable graph and C a simple circuit in G. Then G is planar if and only if the bridges $B_1,\ldots,B_k$ of G, with respect to C, satisfy the following conditions:

(i) $C + B_i$ is planar, for each $1 \leq i \leq k$;

(ii) the set of bridges can be partitioned into two sets such that no two bridges in the same set interlace.

Proof: See, for example, Fisher and Wing (1966), and Tutte (1960).

To determine whether the required partition (ii) exists, we construct the auxiliary graph A(C) of G by:

(i) $a_i \in V(A)$ iff $B_i \subset G$,

(ii) $(a_i,a_j) \in E(A)$ iff $B_i$ and $B_j$ interlace (with respect to C) in G.

Then the partition exists iff A(C) is bipartite.

(One of the advantages of the DFS-tree structure employed by Hopcroft and Tarjan for the non-updating case is that this partitioning test may be carried out very simply using a three-stack mechanism.)

Hence, the basis for planarity testing is

(i) determine C,

(ii) construct the bridges $B_i$ with respect to C,

(iii) test for the existence of a partition of the bridges into the outside or inside of a planar embedding of C,

(iv) if such a partition exists, check the planarity of each $C + B_i$.

We now turn to the development of a procedure for handling general graphs, i.e.: a method that makes full use of the updating nature of the problem, without the need for the initial circuit to be Hamiltonian, as was the case for GREEDY_HAMILTONIAN in Chapter 4.

For ease of exposition, we choose to use the ideas and nomenclature of Fisher and Wing (1966). Although not a linear-time algorithm, the Fisher and Wing (FW) method is intuitively appealing, and does not depend so intimately upon judicious data representation as, for example, the method of Hopcroft and Tarjan.

In order to outline FW, some notation is required which specialises the definition of bridges. We say that a bridge $B_i$, with respect to a circuit C, denoted $B_i(C)$, is of:

(a) Type I if it consists of a single edge (a,b), with $a,b \in V(C)$,

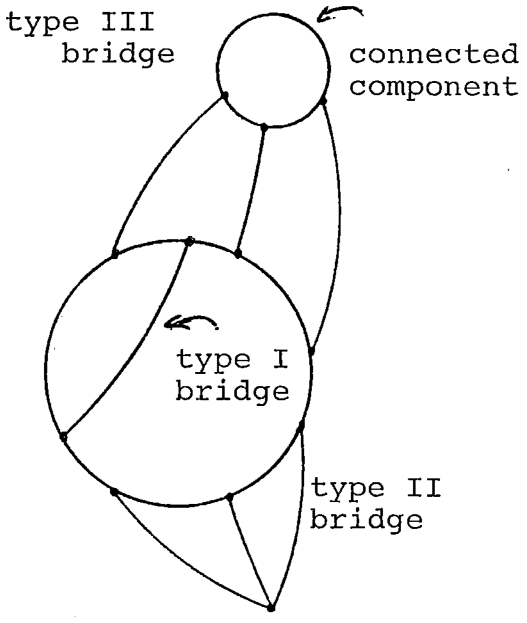
(b) Type II if $|V(B_i(C))| = 1$ and $B_i(C)$ has at least two vertices (edges) of attachment to (in) C,

(c) Type III if $|V(B_i(C))| > 1$ (i.e.: $B_i(C)$ is a connected component) and $B_i(C)$ has at least two vertices of attachment in C.

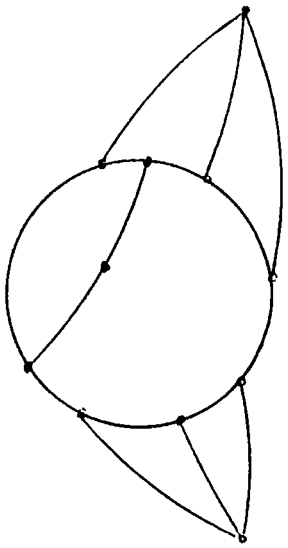
Inherent in definitions (b) and (c) is the fact that we now restrict the vertex set of a bridge to those vertices not contained in C; these remain as vertices of attachment only.

If the vertices of each Type III bridge $B_i(C)$ of G are identified with a single vertex, the resultant graph is termed the reduced graph of G, denoted R(G). (Fisher and Wing call their corresponding graph pseudo-Hamiltonian.)

Figures 5.1 (a), (b) illustrate these definitions.




5.1 (a) Graph G relative to seed circuit C.



5.1 (b) The Reduced Graph of G, R(G).

Figure 5.1

Procedure PLANAR_TEST (as described in Pidgin Pascal in Figure 5.2) outlines the FW method. The essential steps are as follows

(i) determine a "long" circuit, C, through G (the longer the circuit, the less complex the bridges).

(ii) determine the bridges $B_i(C)$ (this may be achieved by construction); form the reduced graph R(G).

(iii) test R(G) for planarity (equivalent to testing for the existence of a bridge partition).

(iv) if R(G) planar then test the planarity of $C + B_i(C)$, for each type III bridge $B_i(C)$, by successive reduced graphs for each until no more type III bridges remain with respect to the 'final' circuit. (It can be shown that this process must converge by ensuring that at least one type I bridge exists with respect to each circuit.) This decomposition approach effectively reduces the planarity testing to seeking partitions of bridges for ever-simplifying reduced graphs.

(v) if any bridge partitioning attempt fails in (iv), then G must be non-planar, STOP. Otherwise, G is planar.

Note that the partitioning test for each reduced graph may be implemented efficiently using a generalisation of the updating strategies of Chapter 4. We may now define the auxiliary graph A(C) by

```
Procedure PLANAR_TEST; (FISHER AND WING)
    Procedure TEST(G');
    begin
        determine "long" circuit C* in G';
        determine bridges B_j of G' relative to C*;
        construct A(C*) via interlacings;
        if A(C*) bipartite then
        begin
            K(C*) := [type III bridges of G' relative to C*];
            for each bridge B_r ∈ K(C*) do TEST(C* + B_r);
        end
        else PLANAR := FALSE;
    end;
begin (* mainline *)
    determine "long" circuit C thru G;
    determine bridges B_j of G relative to C;
    construct A(C) via interlacings;
    if A(C) bipartite then
    begin
        K := [type III bridges of G relative to C];
                (* B_1,...,B_|K| *)
        if |K| > 0 then
        begin
            PLANAR := TRUE; k := 1;
            while (K <= |K| and PLANAR) do
            begin
                TEST(C + B_k);
                k := k + 1;
            end;
```

```
            if (k = |K| + 1 and PLANAR) then output G PLANAR
                                        else output G NON-PLANAR;
        end
        else output G PLANAR:
    end;
end;
```

Figure 5.2

(i) $a_i \in V(A)$ iff $B_i \subset G$

(ii) $(a_i,a_j) \in E(A)$ iff $B_i$ and $B_j$ interlace (with respect to C) in G.

(To ease handling of type I bridges under the definition of interlacement we may transform each into a type II bridge comprising a dummy vertex and two edges of attachment.)

Rapid detection of an induced odd circuit in A (in GREEDY leading to a rejection of the current candidate edge in G) is then possible using the manner described previously, with bridges replacing edges, and bridge interlacings replacing edge crossings.

We now describe how the FW procedure can form a basis for planarity testing in an updating environment.

Consider a planar graph $G = (V(G),E(G))$,

where $V(G) = \bigcup_i V(K_i)$

and $E(G) = \bigcup_i E(K_i)$,

and the $K_i$'s are termed clusters, where a cluster may take any one of four forms:

(1) a connected component of G containing no circuit,

(2) a connected component of G containing a circuit, and no separating edge (the first circuit created in such a

cluster is called the seed circuit)

(3) two type (1) or type (2) clusters joined by a separating edge (the two subclusters are said to be attached),

(4) two type (1) or type (2) clusters joined at their single common vertex (again, the subclusters are attached).

(Note that, from the point of view of planarity, each unit of a type (3) or (4) cluster may be treated independently; the recursive definition is introduced to maintain consistency.)

Consider the addition of a new edge (a,b) to a graph G with the above representation. There exist six distinct forms of addition:

(i) $a,b \notin V(G)$

(ii) $a \in V(K_i), b \notin V(G)$, for some i,
or $a \notin V(G), b \in V(K_i)$, for some i

(iii) $a,b \in V(K_i)$, for some i

(iv) $a \in V(K_i), b \in V(K_j)$, for some $i \neq j$, with $K_i, K_j$ not attached (with respect to each other)

(v) $a \in V(K_i), b \in V(K_j)$, for some $i \neq j$, with $K_i, K_j$ attached directly (i.e.: $(K_i, K_j)$ represents either a type 3 or type 4 cluster),

(vi) $a \in V(K_i), b \in V(K_j)$, for some $i \neq j$, with $K_i, K_j$ attached via a chain of clusters (this can perhaps be most conveniently thought of as a chain of type 2 clusters attached path-wise).

In cases (i), (ii), and (iv), the resultant graph, G + (a,b) must be planar. Otherwise, the application of a planarity testing phase (in the form of algorithm PLANAR_TEST (FW)) is generally required to determine whether the addition is acceptable.

Each of the six cases will be treated in more detail, indicating the main steps required for implementation, and the necessary changes to the underlying data structures. Most notation used is mnemonic or self-explanatory. Updating/ restoration mechanisms for edge acceptance/rejection are assumed to exist implicitly.

```
(i)  if a,b ∉ V(G) (* G + (a,b) planar *)
     then begin
          cluster_no := cluster_no + 1;
          k := cluster_no;
          create new cluster K_k := ({a,b},(a,b)):
          circuit(k) := FALSE;
          attach(k) := FALSE;
          chain_attach(k) := FALSE;
     end;
(ii) if (a ∈ V(K_i), b ∉ V(G))
     or (a ∉ V(G), b ∈ V(K_i))
     then begin
          (* G + (a,b) planar *)
          (* consider case a ∈ V(K_i), b ∉ V(G) *)
          if a ∈ V(C_i) (* seed circuit *)
          then begin
               k := cluster_no := cluster_no + 1;
               create new cluster K_k = ({b},φ);
               circuit(k) := FALSE;
               attach(k) := TRUE;
               if attach(i) = FALSE then attach(i) := TRUE;
               if attach(i) = TRUE
               then begin
```

```
                chain_attach(i) := TRUE;
                chain_attach(i) := TRUE;
                update chain elements;
            end;
            note cluster attachment vertices a*_i, b*_k;
        end;
        if a ∈ V(B(C_i)) (* of type II *)
        then set B(C_i) to type III;
        (* if a ∈ V(B(C_i)) (of type III) then no
              changes *)
        (* if circuit(i) = FALSE then no changes *)
    end;
(iii) if a,b ∈ V(K_i)
    then begin
        if circuit(i) = FALSE (* type 1 cluster *)
        then begin
            (* G + (a,b) planar *)
            E(K_i) := E(K_i) + (a,b);
            (* addition of (a,b) must create seed
                 circuit in K_i *)
            circuit(i) := TRUE;
            set up A(C_i), auxiliary graph relative
                 to C_i;
            (* A contains no edges - no interlacing
                 yet *)
            for each pendant edge (a',b') (* a' ∈
                 V(C_i) *) do as for case (ii);
            for each pendant chain <a',b'...> do set
                 up cluster ({b',...}, ( )) then
```

```
                continue as for case (ii), with
                b ≡ cluster;
        end;
        if circuit(i) = TRUE (* several possibilities
              exist *)
        then begin
            if a,b ∈ V(C_i) (* seed circuit *)
            then begin
                (a,b) forms type I bridge relative
                      to C_i;
                convert to type II bridge; update
                      A(C_i);
                check interlacing (* incrementally *)
                      in A(C_i);
                if A(C_i) remains bipartite
                then PLANAR := TRUE, UPDATE
                else reject (a,b), RESTORE;
            end;
            if a ∈ V(C_i), b ∈ V(B_j(C_i))
            then begin
                (* or, similarly, a ∈ V(B_j(C_i)),
                      b ∈ V(C_i) *)
                if B_j(C_i) type II bridge
                then begin
                    check induced interlacings in A(C_i);
                    if A(C_i) bipartite
                    then PLANAR := TRUE, UPDATE
                    else reject (a,b), RESTORE:
                end
```

```
            else begin (* B_j(C_i) type III *)
            if b = only attachment vertex of B_j(C_i)
            then begin
                check induced interlacings in A(C_i);
                if A(C_i) bipartite
                then PLANAR := TRUE, UPDATE
                else reject (a,b), RESTORE
            end
            else begin
                check induced interlacings in A(C_i);
                if A(C_i) bipartite then
                    test planarity of B_j(C_i) +
                          C_i + (a,b);
                    (* using FW as a subroutine *)
                    if PLANAR = TRUE then UPDATE
                        else reject (a,b), RESTORE
                else reject (a,b), RESTORE;
            end;
        end;
    end;
    if a,b ∈ V(B_j(C_i)) (* B_j(C_i) type III *)
    then begin
        (* no new interlacements relative to C_i *)
        test planarity of B_j(C_i) + C_i + (a,b)
              (* FW *)
        (* if B_j(C_i) has no circuit then planar *)
        if PLANAR = TRUE then UPDATE
            else reject (a,b), RESTORE;
    end;
```

```
if a ∈ V(B_j(C_i)) and b ∈ V(B_k(C_i))
then begin
    (* no new interlacements relative to C_i *)
    if (B_j(C_i) type II and B_k(C_i) type II)
    then begin
        PLANAR := TRUE;
        amalgamate bridges;
        modify combined interlacements;
        (* new bridge is type III *)
    end
    else begin
        test planarity of B_j(C_i) + B_k(C_i) +
              C_i + (a,b);
        if PLANAR = TRUE
        then begin
            amalgamate bridges;
            combine interlacements;
        end
        else reject (a,b), RESTORE;
    end;
end;
(* a possible filter:
    if B_j and B_k are connected by a path in
          A(C_i)
    and colour(B_j) ≠ colour(B_k)
    then planar := FALSE *)
end;
```

```
(iv)  if a ∈ V(K_i), b ∈ V(K_j), i ≠ j, and K_i,K_j not
          attached (with respect to each other)
      then begin
          (* G + (a,b) PLANAR *)
          if attach(i) = TRUE
          then chain_attach(j) := TRUE
          else if attach(j) = TRUE
              then chain_attach(i) := TRUE
          else attach(i) := attach(j) := TRUE;
          (* (K_i,K_j) now represent a type 3 cluster *)
          a*_i := a; b*_j := b; (* cluster attachment
                vertices *)
          update cluster chain; (* if required *)
      end;

 (v)  if a ∈ V(K_i), b ∈ V(K_j), i ≠ j, K_i,K_j attached
          directly (* (K_i,K_j) type 3 or type 4
          cluster *)
      then begin
          if a = a*_i then
              if (K_i,K_j) type 3 cluster
              then begin
                  determine "long" path thru K_j from
                        b to b*_j;
                  complete to circuit C* using edges
                        (b*_j,a), (a,b);
                  test planarity of K_j + (b*_j,a) + (a,b)
                        relative to C*;
                  (* using FW, independently of K_i *)
```

```
            if PLANAR = TRUE then
                UPDATE, set (K_i,K_j) to type 4
                else reject (a,b), RESTORE;
        end
        else (* (K_i,K_j) type 4 cluster *)
            test planarity of K_j + (a,b) relative
                 to C_j
                (* C* as generated above *)
            (* i.e. as for case (iii), a,b ∈ V(C_j) *)
if b = b*_j then (* analogously to a = a*_i *);
if (a <> a*_i and b <> b*_j) then
    if (K_i,K_j) type 3 cluster
    then begin
        determine "long" path C* thru' K_i
              and K_j using edges (a,b),
              (a*_i,b*_j);
        determine bridges relative to C*;
        create V(A(C*));
        if all bridges of types I, II
        then begin
            check interlacings relative to C*;
            if A(C*) bipartite
            then begin
                PLANAR := TRUE;
                UPDATE: (* new cluster K_ij,
                      type 2 *)
            end
            else reject (a,b), RESTORE
        end
```

```
                else begin (* have type III bridges
                      relative to C* *)
                    check interlacings relative to C*;
                    for each type III bridge B_j(C*) do
                    begin
                        check planarity of C* + B_j;
                              (* FW *)
                        if PLANAR = FALSE for any
                              C* + B_j
                            then reject (a,b), RESTORE;
                    end;
                    if PLANAR = TRUE for all C* + B_j
                        then UPDATE; (* new cluster
                              K_ij, type 2 *)
                end;
                (* if PLANAR, also must update cluster
                      chains relative to the new
                      cluster, K_ij *)
            else begin (* (K_i,K_j) type 4 cluster *)
                determine "long" path C* thru K_i and
                      K_j using edge (a,b) and passing
                      thru vertex a_i* (= b_j*);
                continue as for type 3 cluster;
            end;
        end;
```

(vi) $a \in V(K_i)$, $b \in V(K_j)$, $i \neq j$, $K_i, K_j$ attached via a chain of clusters.

(* a modification of case (v), with the "long" path now defined thru the cluster chain, however, many combinations of cluster attachment incidences

may occur. For example, suppose $\langle K_i, K_j, K_r \rangle$ is a cluster chain with cluster attachment vertices $a_i^*, b_i^*, a_j^*, b_j^*, a_r^*, b_r^*$, as depicted in Figure 5.3.

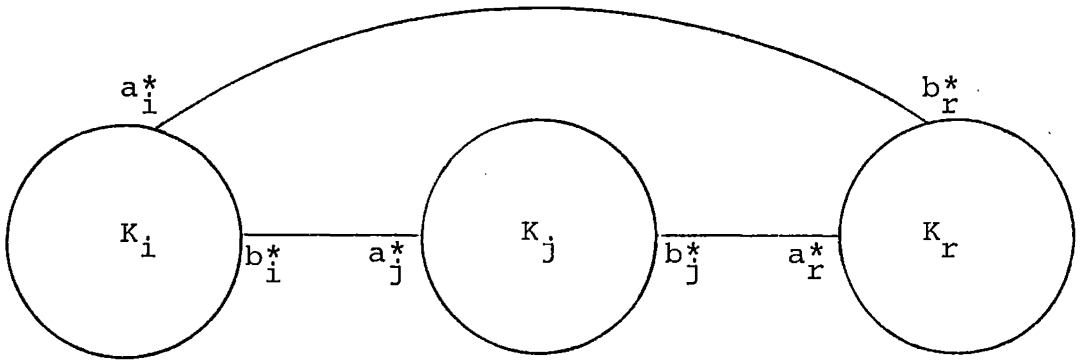



Figure 5.3

(I) $a_i^* = b_i^*$, $a_j^* = b_j^*$, $a_r^* = b_r^*$
each cluster remains mutually independent of the other two (* extension of definition of type 4 cluster *)

(II) $a_i^* <> b_i^*$, $a_j^* = b_j^*$, $a_r^* = b_r^*$ (* wlog *)
edge additions to $K_j$ or K remain independent
extend $K_i$ to include path $\langle b_i^*, a_j^*(=b_j^*), a_r^*(=b_r^*), a_i^* \rangle$

(III) $a_i^* <> b_i^*$, $a_j^* <> b_j^*$, $a_r^* = b_r^*$ (* wlog *)
amalgamate $K_i, K_j$; determine "long" paths $\langle a_i^*, \ldots, b_i^* \rangle$ (thru $K_i$) and $\langle a_j^*, \ldots, b_j^* \rangle$ (thru $K_j$), and extend using $\langle b_j^*, a_r^*(=b_r^*), a_i \rangle$.
edge additions to K remain independent.

(IV) $a_i^* <> b_i^*$, $a_j^* <> b_j^*$, $a_r^* <> b_r^*$

amalgamate $K_i, K_j, K_r$;

determine "long" paths $<a_i^*,\ldots,b_i^*>$ in $K_i$

$<a_j^*,\ldots,b_j^*>$ in $K_j$

$<a_r^*,\ldots,b_r^*>$ in $K_r$

extend using $(b_i^*,a_j^*),(b_j^*,a_r^*),(b_r^*,a_i^*)$ *)

In the preceding discussion we have introduced the concept of a "long path". This is because the efficiency of the FW decomposition approach is improved by the use of seed (and auxiliary) circuits composed of a high number of edges (resulting in reduced complexity of their associated bridges). We suggest the following method for "long" path determination from vertex v to vertex 2 (in a type 1 or type 2 cluster);

let deg(w) := degree of w in $K_i$; avoid_w := 0;

initiate a depth-first-search of $K_i$, rooted at v

using an adjacency list representation adj_list(*)

of $K_i$

while avoid_w< deg(w) - 1 do

if w ∈ adj_list(i) in DFS-tree then

don't visit w, avoid_w := avoid_w + 1;

complete DFS-tree path to w.

As DFS is a linear-time procedure, this modification represents an efficient means of constructing a (v,w)-path of reasonable length.

The efficiency of the above methodology has yet to be proven experimentally. Given the multitude of possible construction scenarios produceable by GREEDY, time-complexity analysis would have to be probalistically determined, but it is felt that the cluster-based technique would perform

favourably when compared to existing start-over procedures.

We mentioned earlier that the Hopcroft and Tarjan procedure is not entirely amenable to allowing an efficient updating form (and similarly for the other linear-time algorithms whose efficiency depends upon initial complete knowledge of the graph). It may well be, however, that even frequent wholesale redefinition of the depth-first-search representation (in the case of Hopcroft and Tarjan) of the graph induces no more of a computational burden than do the redefinitions inherent in our alternative approach - for instance, when determining the structure of the modified bridges with respect to a new long circuit. Certainly the FW procedure is inferior as a subroutine when it is required at the various stages indicated. It would be interesting to determine the relative merits of the depth-first-search tree modification rationale versus that of the cluster-updating method - this should be a worthwhile problem to pursue in future work.

# CHAPTER 6

# MINIMIZING TRANSPORTATION COST

In Chapters 3 and 4 we introduced techniques for constructing highly weighted and adjacency graphs which maximized the sum of relationship chart scores for adjacent or nearly-adjacent pairs of facilities. Our previous definition of the distance between facilities was in terms of unit boundaries - no variations according to facility area were considered. A more realistic model would be to incorporate the desired areas into the formulation to devise an improved distance measure, and use this as a basis for determining the transportation cost between pairs of facilities. This leads to a new problem with the objective of minimizing transportation cost within a layout, given data for travel costs and the expected number of journeys between pairs of facilities. Much of what follows may be found in Foulds and Giffin (1983).

## 6.1 The SUPER_DELTAHEDRON Heuristic

Firstly we define the revised problem, MIN_TRANS, for the minimization of transportation cost in block plan design.

Let n = number of facilities

$w_{ij}$ = product of the cost per unit distance travelled and the number of trips per time period between facilities i and j

$d_{ij}$ = distance travelled between facilities i and j (dependent on the final layout and the distance metric chosen)

$a_i$ = area of facility i

$A = (a_i)_{1\times n}$

L(A) = set of feasible block plans

We assume that

(i) facility shapes are not specified, but should be rectangular or L-shaped where possible.

(ii) all walls are to be parallel to the building boundary.

The objective of problem MIN_TRANS is to

$$\text{MINIMIZE } T(L) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} d_{ij} \quad ;$$

we seek an adjacency graph, G, corresponding to a layout, L, with a relatively low cost, T(L).

Heuristic DELTAHEDRON is again chosen for modification. Initialisation takes the form of either INITIAL_3 or INITIAL_2 as described in Chapter 3. Vertex insertion order is either fixed via the column-sum definition (procedure FIXED_ORDER) or greedily (procedure GREEDY_ORDER). The criterion for the best triangle in the insertion step at the $(k+1)^{st}$ iteration is the one that leads to the minimization of

$$\sum_{y \in V_k} w_{xy} \hat{d}_{xy} \qquad (*)$$

where $T_k = (V_k, E_k)$ is the adjacency graph at the $k^{th}$ iteration

and x = the new vertex being inserted.

The distance $\hat{d}_{xy}$ is an estimate of $d_{xy}$, and is calculated as follows:

Let the shortest path from vertex x to vertex y be given by

$$SP(x,y) = \langle x, v_1, v_2, \ldots, v_m, y \rangle,$$

where $v_1, \ldots, v_m \in V_k$.

Then the <u>length</u> of SP(x,y) is taken to be

$$\hat{d}_{xy} = \tfrac{1}{2}a_x^{\frac{1}{2}} + \sum_{i=1}^{M} a_{v_i}^{\frac{1}{2}} + \tfrac{1}{2}a_y^{\frac{1}{2}},$$

since we assume that each facility j will be a square of side length $a_j^{\frac{1}{2}}$ in the final layout, and that all travel will be rectilinear between facility centroids. Note that this approximation is designed to provide only a ranking among candidate vertex-triangle choices at the insertion stage (*), and may not represent <u>actual</u> distances in the layout.

The simple form of equation (*) is a result of the property of the insertion process taken advantage of in Chapter 4: existing shortest-path lengths in $T_k$ are not affected by the insertion of vertex x, and so need not be considered in the choice. We again use information from each previous iteration to improve the efficiency of the shortest-path calculations at each stage, as follows:

Let x be a candidate for insertion in triangle $(p,q,r) \in T_k$. Then

$$\hat{d}_{xp} = \tfrac{1}{2}(a_x^{\frac{1}{2}} + a_p^{\frac{1}{2}})$$

$$\hat{d}_{xq} = \tfrac{1}{2}(a_x^{\frac{1}{2}} + a_q^{\frac{1}{2}})$$

$$\hat{d}_{xr} = \tfrac{1}{2}(a_x^{\frac{1}{2}} + a_r^{\frac{1}{2}}) \quad .$$

For each vertex $s \in V_k - \{p,q,r\}$,

$$\hat{d}_{xs} = \text{MIN}(\hat{d}_{xp} + \hat{d}_{ps}, \hat{d}_{xq} + \hat{d}_{qs}, \hat{d}_{xr} + \hat{d}_{rs})$$

where the values $\hat{d}_{ps}$, $\hat{d}_{qs}$, $\hat{d}_{rs}$ have all been established at the earlier insertion of vertex s.

In procedure FIXED_ORDER, (*) is evaluated for each of the 2k + 4 triangles in $T_k$, for a given new vertex x. For GREEDY_ORDER we perform the FIXED_ORDER calculation for each vertex x not yet added to $T_k$. The complete procedure, in either of its forms, we term SUPER_DELTAHEDRON.

Pidgin PASCAL descriptions of procedures FIXED_ORDER and GREEDY_ORDER are contained in Figures 6.1 and 6.2. FIXED_ORDER and GREEDY_ORDER are of time complexity $0(n^3)$ and $0(n^4)$ respectively.

## 6.2 Computational Results

In Chapter 4 we discussed the "umbrella" effect. If unconstrained, SUPER_DELTAHEDRON also exhibits the tendency to produce an adjacency graph with at least one vertex of high valency. We therefore again restrict the maximum degree of a vertex in the final solution to an arbitrary level proportional to its area. Again, perturbations may be required in order to obtain a feasible solution under these additional constraints.

Test problems for n = 10,20,30,40 and 50 were generated, again using the method of Box and Muller (1958). For all problems the mean cost was set at 100, with standard deviations ranging from 50 to 250; negative values were reassigned to zero. Facility sizes (in terms of side length) were chosen

```
Procedure FIXED_ORDER;
Procedure UPDATE_SP(x,j,k,r);
begin
    for v ∈ {j,k,r} do
    begin
        SP_length(x,v) := ½*(a_x^½ + a_v^½);
        SP_length(v,x) := SP_length(x,v);
    end;
    for v ∈ V(G) - {j,k,r} do
    begin
        SP_length(x,v) := MIN(SP_length(x,j) + SP_length(j,v),
                              SP_length(x,k) + SP_length(k,v),
                              SP_length(x,r) + SP_length(r,v));
        SP_length(v,x) := SP_length(x,v);
    end;
end;
begin (* mainline *)
    (* initialisation : INITIAL_1 *)
    for j := 1 to n do sum(j) := 0;
    for j := 1 to n
        for i := 1 to n do
            sum(j) := sum(j) + w(i,j);
    order(*) := [sum(i) : sum(i) ≥ sum(j) iff i < j];
    V(G) := [order(1) ... order (4)];
    E(G) := [(order (1),order(2)),...,(order(3),order(4))];
    set up triangle set T(G) corresponding to E(G);
    (* assume a suitable hash function mapping order(i)
         onto i *)
    for i := 1 to 4
        for j := i + 1 to 4 do
```

```
        begin
            SP_length(i,j) := ½*(a_i^½ + a_j^½);
            SP_length(j,i) := SP_length(i,j);
        end;
                  4     4
    tot_ben :=   Σ     Σ    SP_length(i,j)*w(i,j);
                i=1  j+i+1
    for i := 5 to n do
    begin (* insertion phase *)
        min_ben := 10^10; (* sufficiently large number *)
        x := order(i);
        for each triangle (j,k,r) in T(G) do
        begin
            ben := 0;
            UPDATE_SP(x,j,k,r);
            for v ∈ V(G) do
                ben := ben + SP_length(x,v)*w(x,v);
            if ben < min_ben then
            begin
                min_ben := ben;
                best_triangle := (j,k,r);
            end;
        end
        V(G) := V(G) + x;
        E(G) := E(G) + [(x,v) : v in best_triangle];
        adjust T(G);
        tot_ben := tot_ben + min_ben;
        UPDATE_SP (x,best_triangle);
    end;
    output E(G), tot_ben;
end;
```

Figure 6.1

```
Procedure GREEDY_ORDER;
Procedure UPDATE_SP(x,j,k,r);
begin
    (* insert code as in FIXED_ORDER *)
end;
begin (* mainline *)
    (* initialisation : INITIAL_2 *)
    w_ij := MAX w_kr;
            k,r
    select k so that triangle (i,j,k) has maximum weight;
    select r to maximize w_ir + w_jr + w_kr;
    V(G) := [i,j,k,r];
    E(G) := [(i,j),(i,k),(i,r),(j,k),(j,r),(k,r)];
    set up T(G) corresponding to E(G);
    (* assume a suitable hash function mapping i→1, j→2,
        k→3, r→4, etc *)
    for i = 1 to 4
        for j := i + 1 to 4 do
        begin
            SP_length(i,j) := ½*(a_i^½ + a_j^½);
            SP_length(j,i) := SP_length(i,j);
        end;
                4    4
    tot_ben :=  Σ    Σ   SP_length(i,j)*w(i,j);
               i=1 j=i+1
    while |V(G)| < n do
    begin
        min_ben := 10^10;
        for x ∉ V(G) do
        begin
            for each triangle (j,k,r) ∈ T(G) do
            begin
```

```
                ben := 0;
                UPDATE_SP(x,j,k,r);
                for v ∈ V(G) do
                    ben := ben + SP_length(x,v)*w(x,v);
                if ben < min_ben then
                begin
                    min_ben := ben;
                    best_triangle := (j,k,r);
                    best_x := x;
                end;
            end;
        end;
        V(G) := V(G) + best_x;
        E(G) := E(G) + [(best_x,v) : v ∈ best_triangle];
        adjust T(G);
        tot_ben := tot_ben + min_ben;
        UPDATE_SP(best_x, best_triangle);
    end;
    output E(G), tot_ben;
end;
```

Figure 6.2

randomly from {6,8,10,12,14}, with the corresponding initial vertex degree limits respectively set at 6, 7, 8, 9, 10.

Table 6.1 gives a sample of the results obtained. The solution score for GREEDY_ORDER represents the better solution achieved by using INITIAL_2 or INITIAL_3 as the initialisation procedure; usually, INITIAL_3 proved superior.

Table 6.1: Performance Comparison:
FIXED_ORDER vs GREEDY_ORDER

| | | FIXED_ORDER | | | GREEDY_ORDER | | |
|---|---|---|---|---|---|---|---|
| n | σ | T | Σ | v | T | Σ | v |
| 10 | 50 | 0.27 | 59083 | 0 | 0.36 | 58791* | 0 |
| 10 | 150 | 0.28 | 90385* | 0 | 0.33 | 99957 | 0 |
| 10 | 250 | 0.23 | 64009* | 0 | 0.35 | 71597 | 0 |
| 20 | 50 | 0.76 | 406570 | 3 | 3.77 | 345890* | 15 |
| 20 | 150 | 0.91 | 445716* | 0 | 2.59 | 524876 | 0 |
| 20 | 250 | 0.47 | 589715 | 0 | 1.99 | 585967* | 0 |
| 30 | 50 | 1.79 | 946446 | 10 | 19.49 | 691408* | 40 |
| 30 | 150 | 1.50 | 1442371* | 0 | 9.74 | 1571415 | 6 |
| 30 | 250 | 3.37 | 1212614* | 26 | 12.46 | 1491030 | 16 |
| 40 | 50 | 3.46 | 2089141* | 10 | 14.10 | 3136164 | 0 |
| 40 | 150 | 4.67 | 2834429 | 8 | 36.84 | 2534269* | 28 |
| 40 | 250 | 1.80 | 2670875* | 25 | 16.06 | 4821385 | 2 |
| 50 | 50 | 3.71 | 3117682 | 28 | 92.79 | 3001863* | 51 |
| 50 | 150 | 6.22 | 5189647* | 9 | 48.66 | 5689719 | 9 |
| 50 | 250 | 8.54 | 5170891* | 38 | 71.03 | 5567319 | 36 |

n = number of facilities

σ = prescribed standard deviation of cost matrix

T = CPU-time in Burroughs B6930 seconds

Σ = solution score

v = number of violations of degree constraints (total of "excess" adjacencies)

* = best solution score

Neither FIXED_ORDER nor GREEDY_order is clearly the better method, although direct comparison is not possible in instances where the amount of degree constraint violation differs. The impact on the solution score of requiring satisfaction of these constraints is vividly shown by examples (40,50) and (40,250), where approximately 50% improvement has been obtained by their violation in the construction (remembering that such violation occurs only in the pursuit of feasibility). Attainment of a feasible solution via repeated perturbation of the constraint limits also affects processing-time, but to an amount that is problem-specific (compare problems (40,150) and (40,250)). Both procedures still execute efficiently, however, with FIXED_ORDER uniformly faster because of its lower complexity.

In the absence of comparable material, it is difficult to empirically assess the quality of the solutions produced by FIXED_ORDER and GREEDY_ORDER, but, given that they are based upon the successful DELTAHEDRON heuristic, we believe their performance to be of a high standard when applied to non-pathological problems. Only an in-depth worst- (or average-) case analysis similar to that shown in Chapter 3 would provide a definitive judgement.

We note that local improvement strategies of a form comparable to those mentioned in Chapter 3 may be employed to advantage. Global re-evaluation of the shortest path matrix must be undertaken when using the edge-replacement routine, but transference of a vertex, x, of degree 3 requires only recalculation of $SP(x,y)$ for each $y \in V - \{x\}$.

For a certain sub-class of adjacency graphs produced by SUPER_DELTAHEDRON, a block plan may be created using the second method to be introduced in the following Chapter.

# CHAPTER 7

# BLOCK PLAN CONSTRUCTION

In this chapter we develop methodology for creating a block plan corresponding to an adjacency graph. Our approach has been designed for implementation on a microcomputer with 64k bytes of storage, so that certain restrictions on problem size and algorithmic sophistication have been made necessary. In particular, the method is appropriate for problems of at most 20 facilities (including the exterior facility, defined to be facility 1) and the adjacency graph constructed for input to the plan definition phase is constrained to lie within a well-defined, but sufficiently flexible, subset of the set of all triangulations.

## 7.1 Previous graph theoretic work in block plan construction

Teague (1970) uses a network representation introduced by Brooks, Smith, Stone and Tutte (1940) for the dissection of a rectangle into squares. He provides a generalisation of the technique to arrangements of arbitrary rectangles with restrictions "introduced where necessary to retain architecturally significant relationships". A plan is represented by a pair of so-called conjugate networks defined in the "horizontal" (x-coordinate) and "vertical" (y-coordinate) senses. Arcs in the network are defined as

oriented rectangle diagonals; nodes correspond to continuous horizontal (vertical) walls comprising more than one facility; arc flows are set equal to the horizontal (vertical) rectangle dimension; each node is assigned a "potential" - the y-coordinate (x-coordinate) of the horizontal (vertical) wall it represents; arcs incident out of each node are ordered to distinguish among symmetries in the case of multiple arcs. Computational experience with this representation comprises an exhaustive search of the possible networks corresponding to rectangles of given order.

Teague also examines a three-dimensional extension, but use of the structure appears limited to a data-base level.

Grason (1970), and Steadman (1976) suggest dividing the actual adjacency graph into a pair of subgraphs, north-south and east-west. Again this assumes an imposed orientation. However, dimensioning of this graph is only achieved by checking all possible realisations corresponding to feasible arrangements - for more than 8 facilities, a heuristic search technique would need to be developed.

Mitchell, Steadman and Liggett (1976) begin with a plan which maintains the required adjacencies, found via a search of their exhaustive catalogue of topologically distinct partitions. Problem size is limited to a maximum of eight facilities. Dimensions are applied via the solution of a set of simultaneous linear equations which specify

(i) proportioning of individual room width to length, or

(ii) actual width and length requirements.

Areal requirements would result in the solution of quadratic equations and are not included. For dimensioning requirements in terms of ranges of acceptable values, linear programming is used to provide either a feasible solution or a pronouncement of infeasibility, under the objective of maximization or minimization of overall plan length, width, perimeter or proportion ratio.

These dimensioning techniques have been applied to numerous case-study problems, and have been found best suited to housing design, where adjacency, orientation and dimensions can usually be clearly specified, and where plan optimization is often desirable.

Gero (1977) suggests the application of dynamic programming as a means of handling more complex objective functions (such as the minimization of heat loss from a building or the minimization of construction cost - both are functions of total area) in the above method. Computational experience with this alternative shows it to be efficient and effective for small problems.

The main drawback of this census approach is the problem of the large numbers of partitions generated - thus far, complete lists are available only for up to ten facilities (Bloch and Krishnamurti (1978)). For larger problems, attempts have been made to reduce the number of partitions created (see, for example, Flemming (1978)) and Gilleard (1978)) by investigating only those partitions which maintain the prescribed adjacencies.

A further method is due to Roth, Hashimshony and Wachman (1982). Again, the adjacency graph is split into two subgraphs, one representing adjacencies through walls

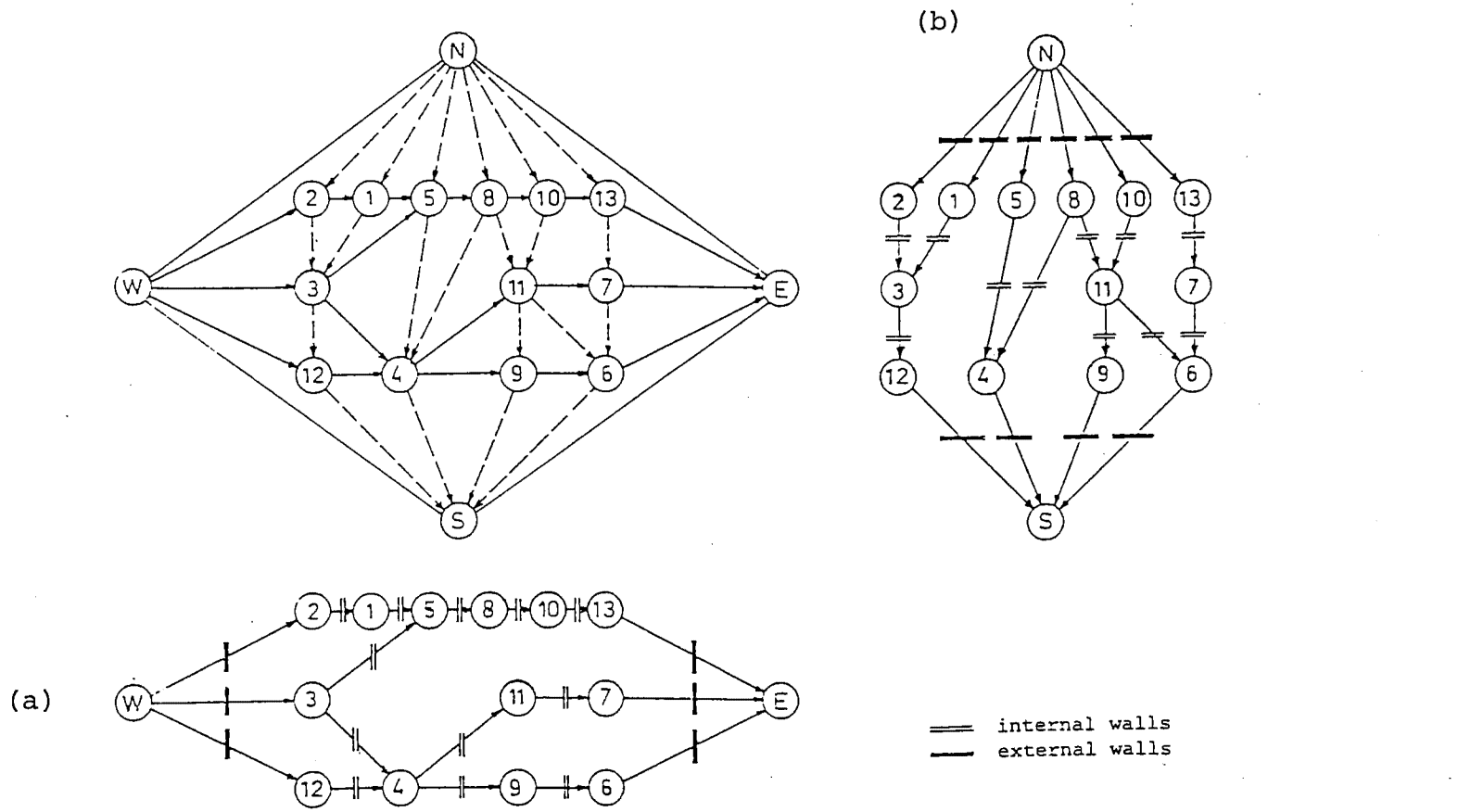



Figure 7.1 The two colour directed subgraphs (CDS's)
(a) CDS representing adjacencies through walls in the y-direction,
(b) CDS representing adjacencies through walls in the x-direction.

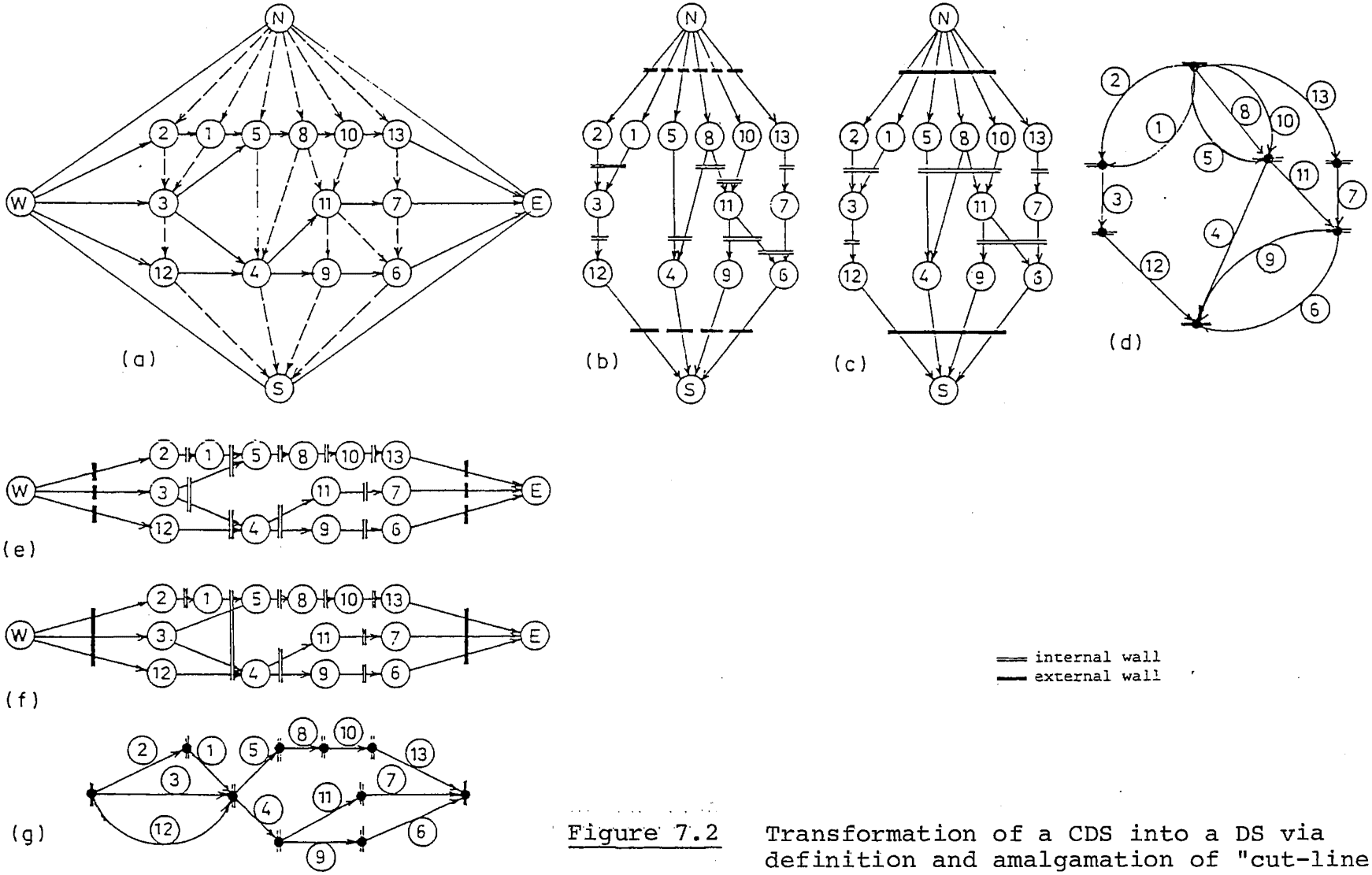



Figure 7.2 Transformation of a CDS into a DS via definition and amalgamation of "cut-lines".

in the x-direction, and the other representing adjacencies through walls in the y-direction. (Adjacencies with respect to north, east, south and west must be provided, as well as maximum and minimum dimensions for each facility.) The splitting technique involves both colouring and directing edges of the adjacency graph; as the specified adjacency graph may not be maximally planar, further edges may be added at this stage to ensure that

(i) any circuit containing one of the four exterior facilities consists of exactly three vertices, and

(ii) any 'interior' circuit contains at most four vertices.

Figure 7.1 gives an example of an adjacency graph and its two coloured directed subgraphs (CDS's). The next step in the procedure unambiguously converts these CDS's into dimensions subgraphs (DS's) whose vertices represent walls and whose weighted edges represent distances between the parallel walls (see March and Steadman (1971) for a similar approach, where the weights represent the actual wall lengths). Figure 7.2 shows the evolution of these DS's from the graphs of Figure 7.1; Figure 7.3 displays the DS's with a set of assigned dimensions. These graphs are treated as flow networks, with source and sink as indicated. It is assumed that the objective is to create a convex plan consisting of convex facilities (of acceptable area) whose perimeter is approximately minimal; the technique used for determining the dimensions of the minimal "envelope" is PERT, (see, for example, Daellenbach, George and McNickle (1983)) which finds the 'critical path' of facilities in each of the x and

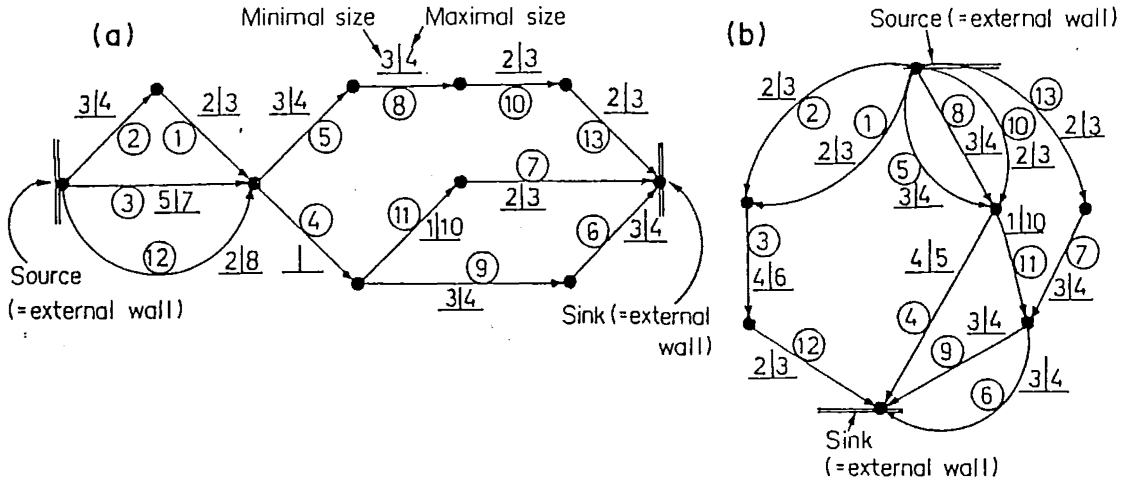



Figure 7.3 Assigning minimum|maximum dimensions to the DS
(a) Distances between y walls
(b) Distances between x walls
(the circled numbers are edge identifiers).

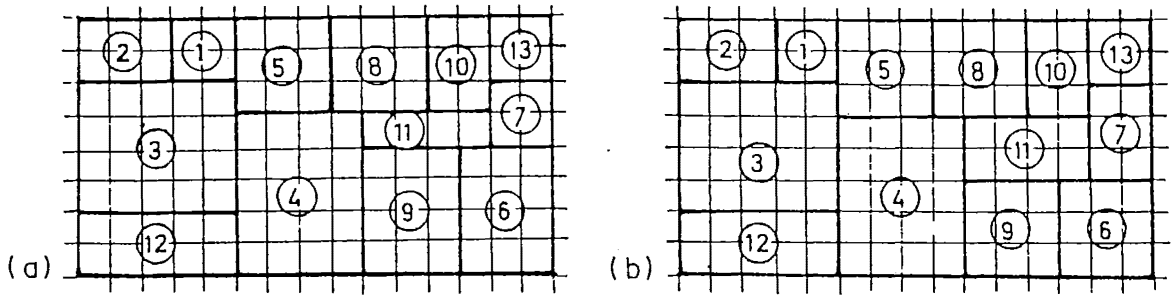



Figure 7.4 Realisation of layout alternatives corresponding to Figure 7.3

y directions - all facilities represented by edges on the critical path are assigned their minimal dimensions.

Translation of the DS's into the corresponding block plan requires checking combinations of the dimensions to find a feasible realisation. The authors state that this may easily be done. Figure 7.4 presents two layout alternatives corresponding to Figure 7.3. The full method appears to be successful in laying out buildings with 'large' numbers of facilities, and has been modified to incorporate non-convex facilities and non-convex perimeters.

## 7.2 The Construction procedure

The procedure is based on a partition of the adjacency graph into sets, $DC_i$, relative to facility 1, according to the edge-distance definition of Chapter 4,

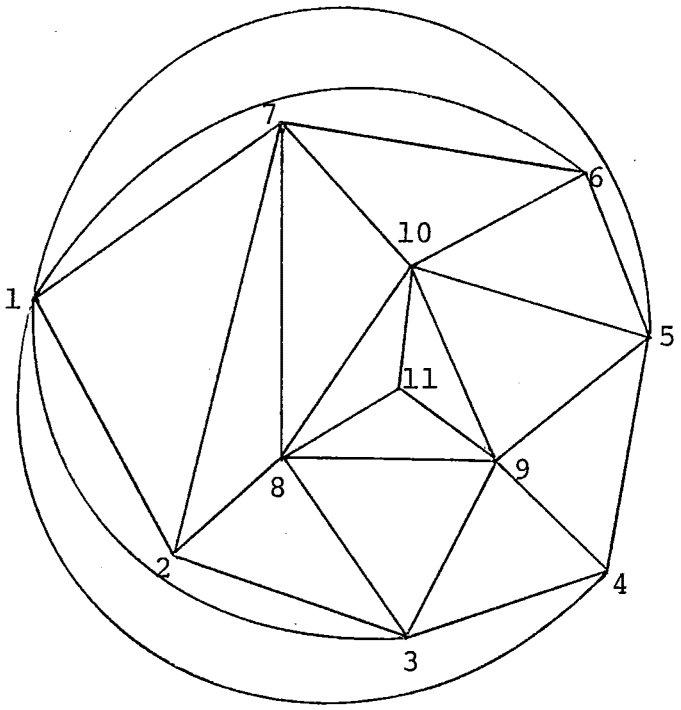
i.e.: $j \in DC_i \iff |SP(j,1)| = i$,

where $|SP(j,1)| = 1$ if the shortest path from vertex j to vertex 1 comprises i edges in G. We call the $DC_i$'s distance classes. In general there will be at most $[\frac{n-2}{3}] + 1$ distance classes for any given graph G, with $|V| = n$.

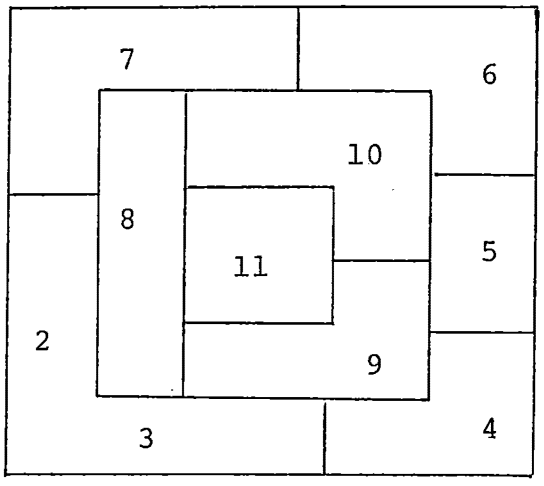
Each set $DC_i$ may be intuitively thought of as representing a "ring" of facilities when translated into a block plan, with each ring nested concentrically. Figure 7.5 displays an example of an adjacency graph and a corresponding feasible plan, showing the ideal situation of ring nesting.

The shortest cycle detectable in distance class $DC_i$ we shall call the ring cycle, R(i).

In Figure 7.5, for instance,




(i) the adjacency graph, G.
G has 3 distance classes:
$DC_1 = \{2,3,4,5,6,7\}$
$DC_2 = \{8,9,10\}$
$DC_3 = \{11\}$




(ii) a corresponding block plan, ignoring areas

Figure 7.5 Ideal ring nesting

R(1) = <2,3,4,5,6,7>

R(2) = <8,9,10>

R(3) = <11>.

Analysis of the construction properties of DELTAHEDRON shows, however, that it is not possible to create an adjacency graph of the form of Figure 7.5, because of the following result.

Theorem 7.1: For each distance class $DC_i$ of an adjacency graph G constructed by the Deltahedron heuristic,

$$|R(i)| \leq 3.$$

Proof: Firstly we prove a simple lemma.

Lemma 7.1: For $i > 1$, the elements of any connected component of a distance class $DC_i$ are wholly contained within a triangle of $DC_{i-1}$.

Proof: For any vertex x, let $d(x) = i \iff x \in DC_i$. Upon insertion of x in triangle (k,m,p), d(x) is defined as

$$d(x) = i, \text{ where } i = MIN\{d(k),d(m),d(p)\} + 1.$$

To create an element of a 'new' distance class relative to triangle (k,m,p), we must have $i \neq d(k), d(m), d(p)$ and hence,

$$d(k) = d(m) = d(p) = i - 1, \text{ necessarily.}$$

Further elements of the connected component of $DC_i$ containing x must be successively added so as to be reachable from x, via members of $D_i$ only and therefore must also be contained within triangle $(k,m,p) \in DC_{i-1}$. QED.

Hence, without loss of generality, we may consider the evolution of a 'new' distance class $DC_i$ through the

successive insertion of vertices $x_1, x_2, \ldots$, say, into a particular triangle of $DC_i$, triangle $(k,m,p)$.

Insert $x_1$ into $(k,m,p)$, creating the triangles $(k,m,x_1)$, $(k,p,x_1)$, $(m,p,x_1)$. Through symmetry we may now insert $x_2$ into any of these triangles. Choose $(k,m,x_1)$. This leads to the situation of Figure 7.6. If we now insert $x_3$ in either of the triangles $(x_1,x_2,k)$ or $(x_1,x_2,p)$, we obtain a cycle of length 3, so the only alternatives are $(k,x_2,p)$, $(k,x_1,m)$, $(m,x_1,p)$. It is clear from Figure 7.6 that a series of insertions in any of these triangles will not produce a cycle of length larger than 3, since there is

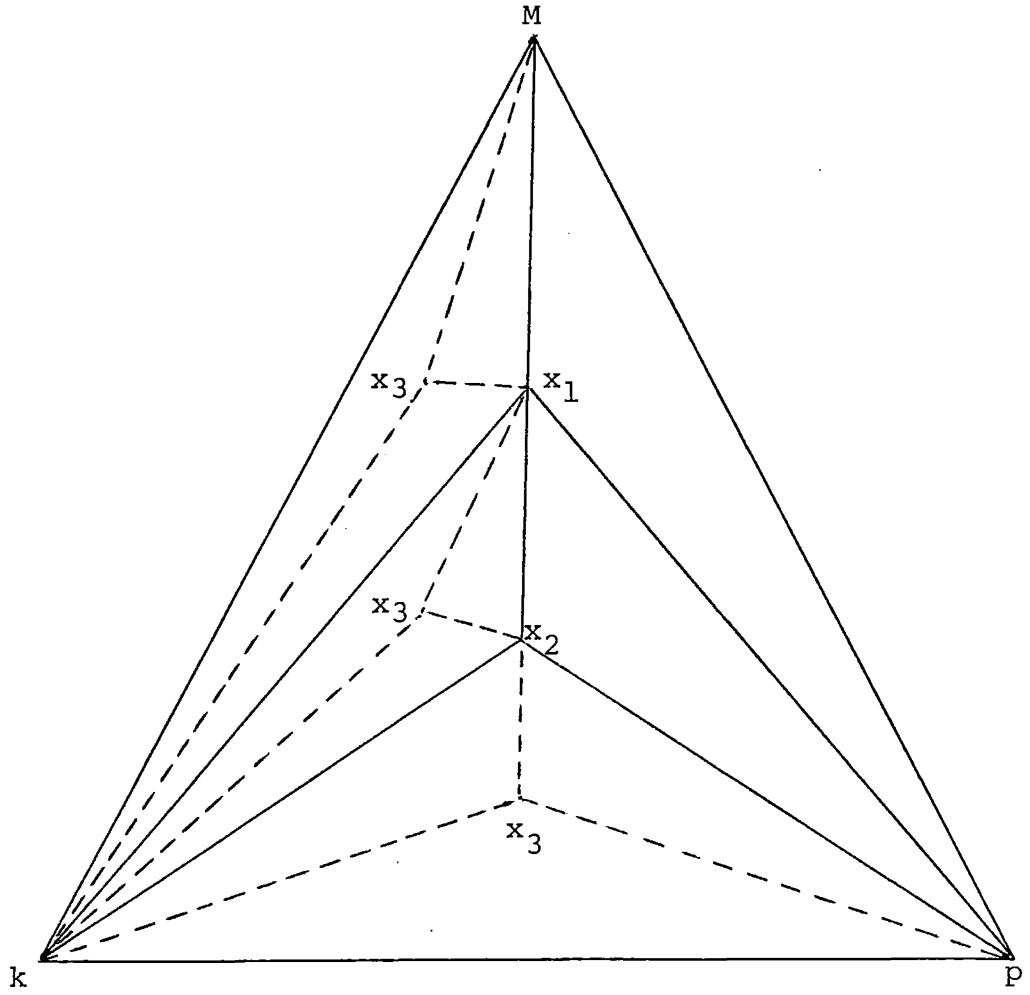



Figure 7.6 Insertion alternatives

no way for a path to return to either $x_1$ or $x_2$ once the first insertion is made. Hence $|R(i)| \leq 3$. QED

The arguments above indicate how we may categorize the insertion process into 3 distinct types according to the distance classes of the members of the insertion triangle: Consider inserting vertex x in triangle (k,m,p), where k,m,p are suitably ordered

TYPE I insertion: $d(k) = d(m) = d(p) = i$

TYPE II insertion: $d(k) = d(m) = d(p) + 1$

TYPE III insertion: $d(k) = d(m) = d(p) - 1$

TYPE I corresponds to creating a new component of $DC_{i+1}$, e.g.: insertion of $x_1$ above

TYPE II corresponds to creating a triangle, (k,m,x) in $DC_{d(k)}$, e.g.: insertion of $x_3$ in $(x_1,x_2,k)$

TYPE III corresponds to augmenting a connected component of $DC_{d(p)}$, e.g.: insertion of $x_2$.

In order to make the block plan construction procedure tractible on a microcomputer, we must enforce some restrictions upon the processes, so that the adjacency graphs obtained fall within a well-defined class of structures. These constraints enable program execution in reasonable time, and reduce the need for complicated overlay programming techniques, while still providing sufficient flexibility to provide practical block plans for moderately-sized problems (of 15-25 facilities).

In particular, the layout procedure as implemented produces plans with the following characteristics:

PROPERTY 1: Only two distance classes are allowed i.e.: a TYPE I insertion is forbidden if $k,m,p \in DC_2$. In addition, each distance class may comprise only one connected component. For many smaller problems these are not overly restrictive assumptions, as few layouts require that a facility be three walls from the exterior, or possess essentially autonomous subgroups of facilities.

PROPERTY 2: In the second distance class, $DC_2$, only one triangle composed of vertices from $DC_2$ is permitted; that is, a TYPE II insertion may occur only once for a triangle comprising elements from $DC_2$. This reduces the complexity of the structure of the inner ring, particularly in cases of facility nesting.

PROPERTY 3: For $DC_1$ apart from the 'fundamental' triangle formed within the first distance class as part of the initial tetrahedron, all insertions made (using TYPE II insertion) into triangles of the form (i,j,1) are done according to a special rule, as follows. Relative to each pair of vertices of the fundamental triangle of $DC_1$, only one 'chain' of insertions is allowed, with each successive TYPE II insertion (after the first) taking place in a triangle containing only one of the vertices of the pair. As an illustration, refer to Figure 7.7, where every new vertex in the chain <a,b,c> between $x_1$ and $x_2$ is adjacent to $x_1$ and 1. Specifically, insertions into (1,a,b), or (1,b,c), or ($x_1$,a,b) etc. are disallowed.

In Chapter 3 it was mentioned that the most effective and efficient rationale for the choice of initial tetrahedron and vertex insertion order was via the column sum approach.

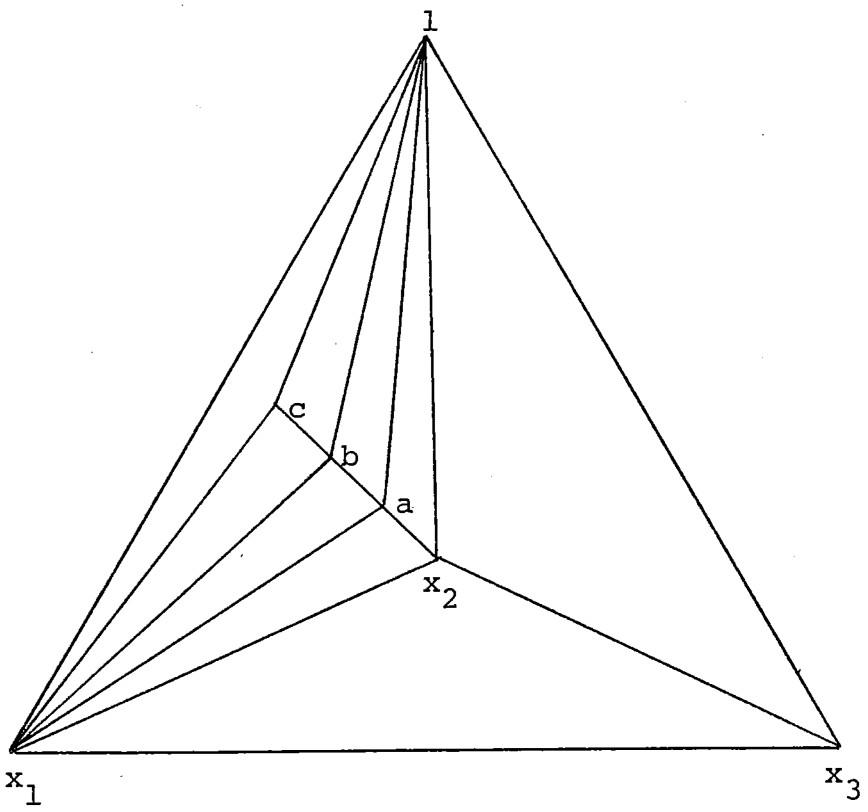



Figure 7.7 Chain of type II insertions

For the restricted approach here we revert to the greedy alternative, since, in order to calculate distances from facilities to the exterior (to determine the composition of the distance classes), it is necessary that the exterior facility form part of the initial tetrahedron.

The initial tetrahedron choice is therefore via the triple {i,j,k} satisfying

$$\underset{i,j,k}{\text{MAX}} \{w(1,i) + w(1,j) + w(1,k) + w(i,j) + w(i,k) + w(j,k)\}$$

Triangle (i,j,k) is then defined as the fundamental triangle for $DC_1$.

Each subsequent vertex insertion is on the basis of the best vertex-triangle pairing in terms of maximizing the increase in objective function value i.e.: choosing vertex x to be inserted into triangle (a,b,c) with w(a,x) + w(b,x) + w(c,x) maximal.

Two phases constitute the generation of a block plan. Firstly the construction of the adjacency graph using PROPERTIES 1-3 with all pertinent structural information; and secondly, the creation of a plan using the output from the first phase as a guideline.

In the case of our restricted adjacency graph, the following information from phase 1 is required for input to phase 2:

(i) the elements and cardinality of $DC_1$ and $DC_2$

(ii) the fundamental triangle, $FT_1$, of $DC_1$

(iii) the fundamental triangle, $FT_2$, of $DC_2$

(iv) the cyclical orientation of $FT_2$ with respect to $FT_1$

(v) for each pair of vertices $x_1, x_2$ of $FT_1$, the (ordered) chain of vertices successively inserted in triangle $(1, x_1, x_2)$ with a flag on either $x_1$ or $x_2$, dependent upon which vertex the chain is adjacent to

(vi) the first vertex inserted in triangle $(x_1, x_2, x_3)$ - called ORIGIN $(DC_2)$ (of distance class 2)

(vii) for each vertex x of $DC_2$ - {ORIGIN$(DC_2)$}, its vertex successor, AFTER(x). If $FT_2$ exists (i.e.: has been created during the construction process), then AFTER(x) is relative to $FT_2$ (i.e.: y = AFTER(x) <=> x is fewer edges from a vertex of $FT_2$); otherwise, AFTER(x) is relative to ORIGIN$(DC_2)$. In this latter case, ORIGIN$(DC_2)$ may have up to three successors; otherwise at most two. The (unique) inverse relationship, BEFORE(x), defined for all

$x \in DC_2 - \{ORIGIN(DC_2)\}$ is also useful.

(viii) if $FT_2$ exists, the length of the path (using elements of $DC_2$ only) from $ORIGIN(DC_2)$ to one of the vertices of $FT_2$.

We term the version of DELTAHEDRON including PROPERTIES 1-3 and only two distance classes constrained DELTAHEDRON.

Figure 7.8 gives an illustration of an adjacency graph constructed using constrained DELTAHEDRON; Figure 7.9 summarizes the information required to represent the structure of this adjacency graph for further processing (See Figure 7.12, later, for the block plan analogies of the terms mentioned).

In order to maintain consistency of adjacency between the two ring cycles, the elements of $FT_1$ and $FT_2$ must be oriented with respect to each other. We may achieve this efficiently as follows.

Consider the order of the elements of $FT_1$ to be fixed, say $FT_1 = \langle x,y,z \rangle$. Without loss of generality, $FT_2$ may be created by a TYPE II insertion of vertex v in triangle (a,b,y), as depicted in Figure 7.10 i.e.: $FT_2 = \langle a,b,v \rangle$.

There are only two possible alternatives for the orientation of $FT_2$, namely $\langle a,b,v \rangle$ or $\langle a,v,b \rangle$. Each is immediately characterised by the position of v, and whether a = BEFORE(b) or b = BEFORE(a), as follows:

Let $FT_i(j)$ be the $j^{th}$ element of $FT_i$

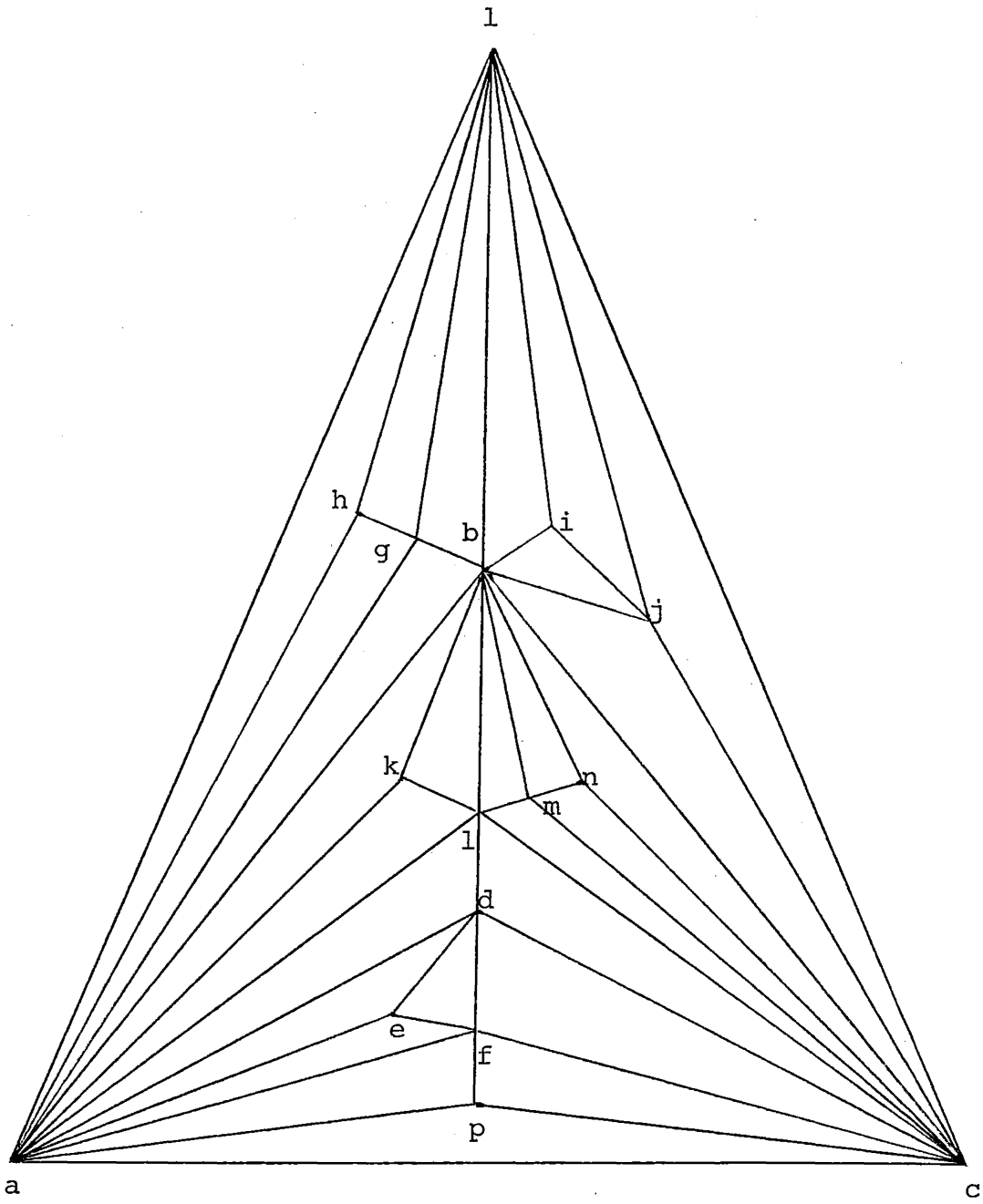



Figure 7.8 Example for illustration of nomenclature

$FT_1$ = (a,b,c)

$FT_2$ = (e,d,f)

r = ORIGIN($D_2$)

AFTER(f,1) = p

AFTER(r,1) = d

AFTER(r,2) = m

AFTER(d,1) = r

AFTER(m,1) = n

BEFORE(f) = d (by construction)

BEFORE(d) = r

BEFORE(k) = r

BEFORE(p) = r

BEFORE(m) = r

BEFORE(n) = m

g,h are between a and b, both adjacent to a

j,i are between b and c, both adjacent to b

TOP = b

LEFT = c

RIGHT = a

INTOP = d

INLEFT = f

INRIGHT = e

length of the shortest path from r to d is 1

Figure 7.9 Data requirements to represent the structure of Figure 7.8

if BEFORE(a) = b then swap a,b

determine the two vertices $x,y \in FT_1$ to which b is adjacent

if $x = FT_1(1)$ and $y = FT_2(3)$ then swap x,y (to maintain cyclic consistency)

if $(v,x) \in E$ then orientation is (a,v,b)

else orientation is (a,b,v)

We may use this oriented cycle in order to specify the actual positions of the elements of $FT_1$ and $FT_2$ in the block plan, but it turns out to be just as efficient to do this by first principles, using only the unordered elements of $FT_1$ and $FT_2$. (In fact, for our restricted problem the orientation procedure is not strictly necessary - it is just an alternative characterisation.)

Suppose $FT_1 = (x,y,z)$

and $FT_2 = (a,b,v)$, as in the example of Figure 7.10.

Then BEFORE(v) = 0 implies INRIGHT = v;

BEFORE(b) = a implies INLEFT = b;

otherwise INTOP = a

$(v,y) \in E$ implies RIGHT = y;

$(b,z) \in E$, $(b,x) \notin E$ implies LEFT = z;

otherwise TOP = x.

The above approach is valid for all configurations allowed in the restricted construction.

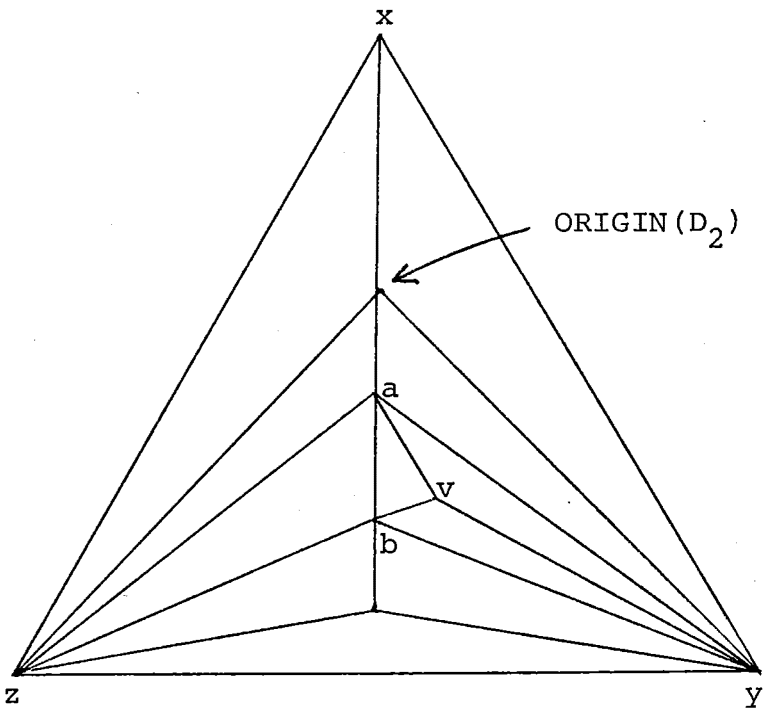



Figure 7.10 Orientation nomenclature

## 7.3 The layout phase

We now describe two plan generation approaches based on the DELTAHEDRON heuristic and the SUPERDELTAHEDRON heuristic incorporating the ideas introduced above.

Firstly, we consider how the information provided in (i) - (viii) above is used by the layout phase in the context of the constrained DELTAHEDRON heuristic.

The general two-distance-class layout may be depicted as in Figure 7.11, under the assumptions

(a) the building exterior is to be regular and rectangular, with width to length ratio specified by a variable, RATIO $\geq$ 1

(b) the shape of the area occupied by the elements of distance class 2 is assumed

congruent to the full building shape. (Note that $DC_2$ may be empty).

We also assume, for the case where the adjacency graph is constructed using the constrained DELTAHEDRON approach (i.e.: facility areas are not explicitly included), that the respective areas of distance classes $DC_1$ and $DC_2$ are proportional to the respective class cardinalities.

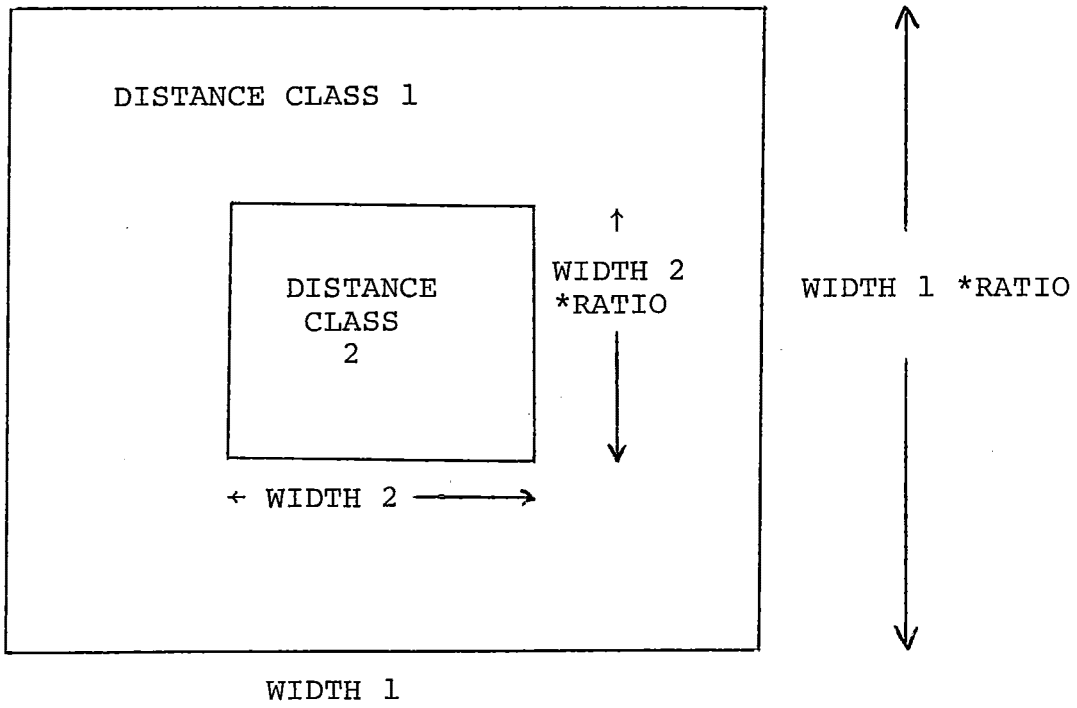



Figure 7.11 The general layout scheme

In the terminology of Figure 7.11 this is equivalent to

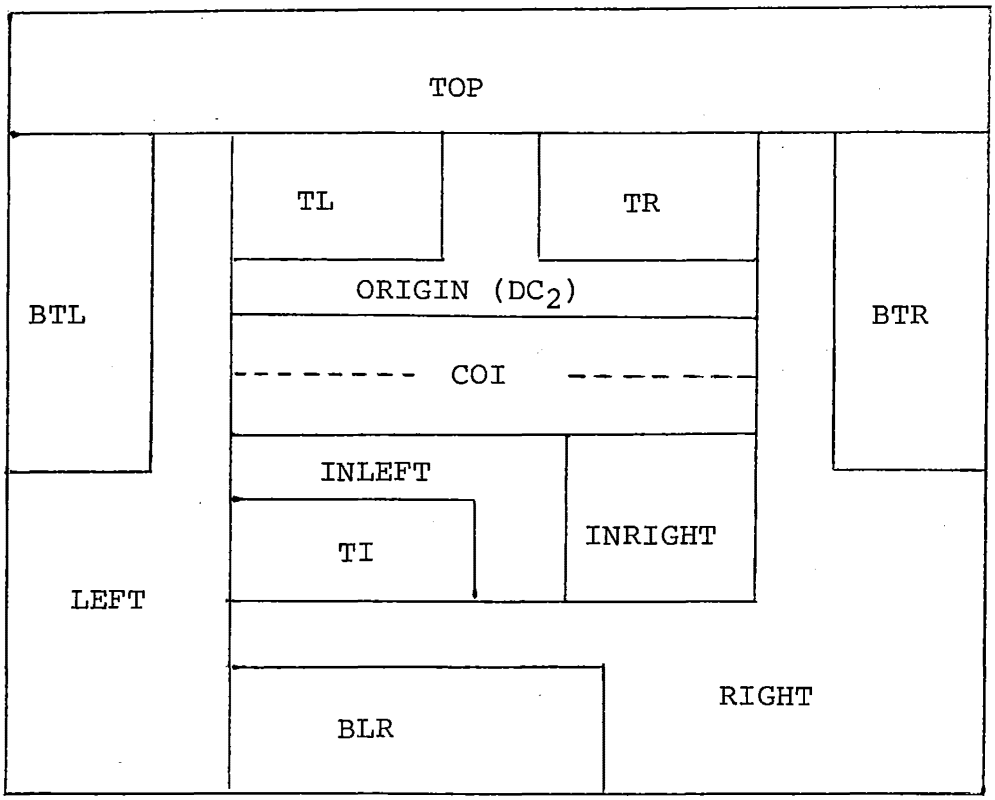
$$\frac{WIDTH2*(WIDTH2*RATIO)}{|DC_2|} = \frac{WIDTH1(*WIDTH1*RATIO) - WIDTH2*(WIDTH2*RATIO)}{|DC_1|}$$

Rearranging yields the condition

$$\frac{WIDTH2*WIDTH2}{WIDTH1*WIDTH1} = \frac{|DC_2|}{|DC_1|+|DC_2|} = \frac{|DC_2|}{n} \quad .$$

Figure 7.12 shows the general form of a layout as produced by constrained DELTAHEDRON. Not all the facets indicated may be present in any one solution; for example, the chain of facilities between ORIGIN($DC_2$) and INLEFT has been observed only infrequently during performance testing. Figures 7.13(a) and 7.13(b) display the structure of the two types of chains that may be constructed between pairs of elements of $FT_1$ (TOP,LEFT,RIGHT) using the restricted form of insertion into triangles including vertex 1 (PROPERTY 3). If the chain contains only one element, the structures are equivalent. The chains emanating from ORIGIN($DC_2$) and INLEFT possess a structure similar to that of Figure 7.13(b). The chain indicated by COI consists of vertices of degree four, each adjacent to both LEFT and RIGHT.

Both constrained DELTAHEDRON and the plan generation phase have been programmed in Microsoft BASIC Version 5.0 and implemented on a Systems Group 2800 microcomputer with 64k RAM. The program listings, DELTAH and PLAN are constrained in Appendix 1. Input may be in the form of a numerical or alphabetic relationship matrix, with a maximum of 30 facilities allowed. As mentioned earlier the user may specify a vertical to horizontal distance ratio ($\geq$ 1) for the plan, and a desired width for the output, whose format is a symbolic encoding for ease of deciphering. Suitable area scaling is provided automatically. Figure 7.15 gives the output from a typical session, using the problem data in Figure 7.14 (after Francis and White (1974)). Total CPU-time required for execution, using the BASIC interpreter only, is approximately 200 seconds; much of that




BTL = chain of facilities between TOP and LEFT (resulting from a sequence of TYPE II vertex insertions involving vertices TOP, LEFT and 1).

BTR, BLR = chains between TOP, RIGHT and LEFT, RIGHT

TL = chain of facilities between TOP and LEFT (from a sequence of TYPE III insertions)

TR, TI = chains between TOP, RIGHT AND LEFT, RIGHT

COI = chain of facilities between ORIGIN ($DC_2$) and INLEFT (occurs if $FT_2$ does not include ORIGIN ($DC_2$) as one of its vertices)

Figure 7.12 General form of a block plan as constructed by constrained DELTAHEDRON

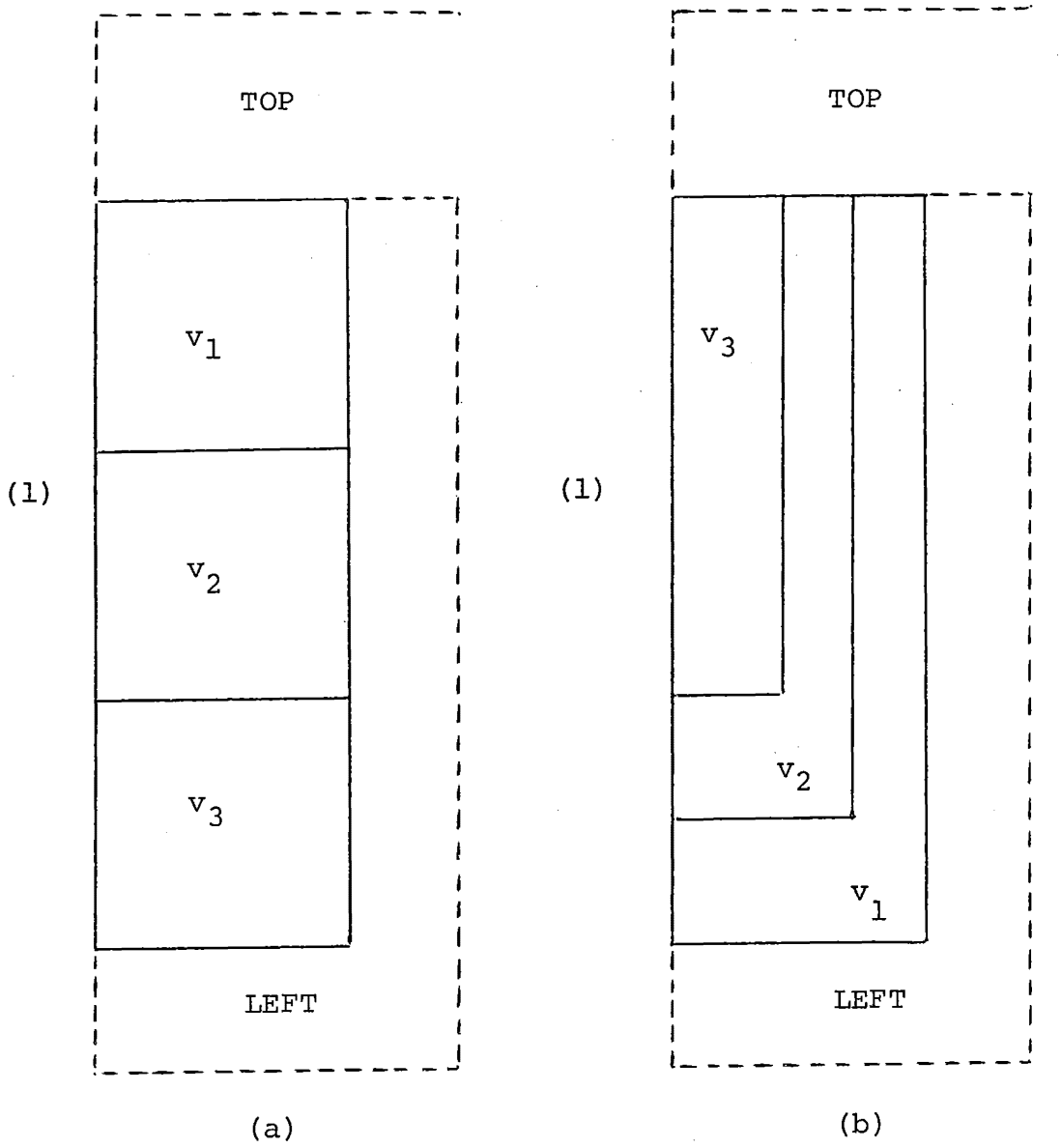



Figure 7.13 Chain of vertices $\langle v_1, v_2, v_3 \rangle$ initiated by a type II insertion in triangle (1,TOP,LEFT)

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | I | I | U | U | A | A | U | U | I | E | A | U |
| 2 | | | I | U | X | X | U | U | U | U | O | O | U |
| 3 | | | | U | U | U | U | I | U | U | U | U | U |
| 4 | | | | | A | I | E | E | U | E | U | I | I |
| 5 | | | | | | U | I | X | U | E | I | I | I |
| 6 | | | | | | | I | X | U | U | I | E | I |
| 7 | | | | | | | | E | I | E | I | I | I |
| 8 | | | | | | | | | E | U | A | U | U |
| 9 | | | | | | | | | | U | E | I | U |
| 10 | | | | | | | | | | | U | O | O |
| 11 | | | | | | | | | | | | U | O |
| 12 | | | | | | | | | | | | | U |
| 13 | | | | | | | | | | | | | |

KEY:

A = absolutely necessary = 64
E = especially important = 16
I = important = 4
O = ordinary closeness = 1
U = unimportant = 0
X = undesirable = -128

1 exterior facility
2 office
3 rest, lunch and recreation
4 welding
5 press
6 foundry
7 machining
8 assembly
9 painting
10 steel storage
11 other storage
12 receiving, shipping, finished stores
13 maintenance

Figure 7.14 Data for example problem

time is taken up with calculating the value of the best possible solution to provide a measure of performance. Note that the achieved solution attained 92.2% of the theoretical maximum (which may not be even achievable); on the same problem, the best score obtained by ALDEP (Seehof and Evans (1967)) was 76% of the best possible.

The plan produced in Figure 7.15 could form the basis for an efficient layout with little alteration, but often the output should be thought of as representing only a framework guideline on which to base the final plan. This is especially the case in the presence of substructures similar to that of Figure 7.13(b) where it may not be convenient to have nested L-shapes. With more than 20 facilities however, it is also likely that such nested chains may become impracticably long when only two distance classes are permitted. When the actual facility areas are also considered, the constrained DELTAHEDRON plan may not even be feasible without the addition of vertex degree constraints. Hence we now turn our attention to the modifications required to the basic block plan construction scheme to rudimentarily incorporate the rationale of SUPERDELTAHEDRON in an attempt to improve the practicality of the block plans produced.

The version of SUPERDELTAHEDRON we use is completely analogous to constrained DELTAHEDRON - PROPERTIES 1-3 still hold as construction restrictions. Vertex degree constraints are provided as described in Chapters 4 and 6. Given the small difference in solution quality of the column-sum insertion order and greedy rationales for SUPERDELTAHEDRON, we use the FIXED_ORDER method in order to reduce the computational burden. We call the restricted heuristic constrained SUPERDELTAHEDRON.

Figure 7.15

```
NUMBER OF FACILITIES: 13

 TYPE OF DATA INPUT REQUIRED?    R

       (Corresponds to reading from data statements)

RELATIONSHIP MATRIX IN THE FORM (I,J), FOR J ) I

 1  : I I U U A A U U I E A U
 2  :   I U X X U U U U O O U
 3  :     U U U U I U U U U U
 4  :       A I E E U E U I I
 5  :         U I X U E I I I
 6  :           I X U U I E I
 7  :             E I E I I I
 8  :               E U A U U
 9  :                 U E I U
 10 :                   U O O
 11 :                     U O
 12 :                       U
 13 :

RELATIONSHIP MATRIX IN NUMERICAL TERMS

 0  132  132  128  128  192  192  128  128  132  144  192  128
 132  0  132  128  0  0  128  128  128  128  129  129  128
 132  132  0  128  128  128  128  132  128  128  128  128  128
 128  128  128  0  192  132  144  144  128  144  128  132  132
 128  0  128  192  0  128  132  0  128  144  132  132  132
 192  0  128  132  128  0  132  0  128  128  132  144  132
 192  128  128  144  132  132  0  144  132  144  132  132  132
 128  128  132  144  0  0  144  0  144  128  192  128  128
 128  128  128  128  128  128  132  144  0  128  144  132  128
 132  128  128  144  144  128  144  128  128  0  128  129  129
 144  129  128  128  132  132  132  192  144  128  0  128  129
 192  129  128  132  132  144  132  128  132  129  128  0  128
 128  128  128  132  132  132  132  128  128  129  129  128  0


BEST TETRAHEDRON:

 1  6  7  12

INSERTING VERTEX 4 IN TRIANGLE 6  7  12

INSERTING VERTEX 5 IN TRIANGLE 4  7  12

ORIENTED CYCLE:

 2 : 4  10  5
```

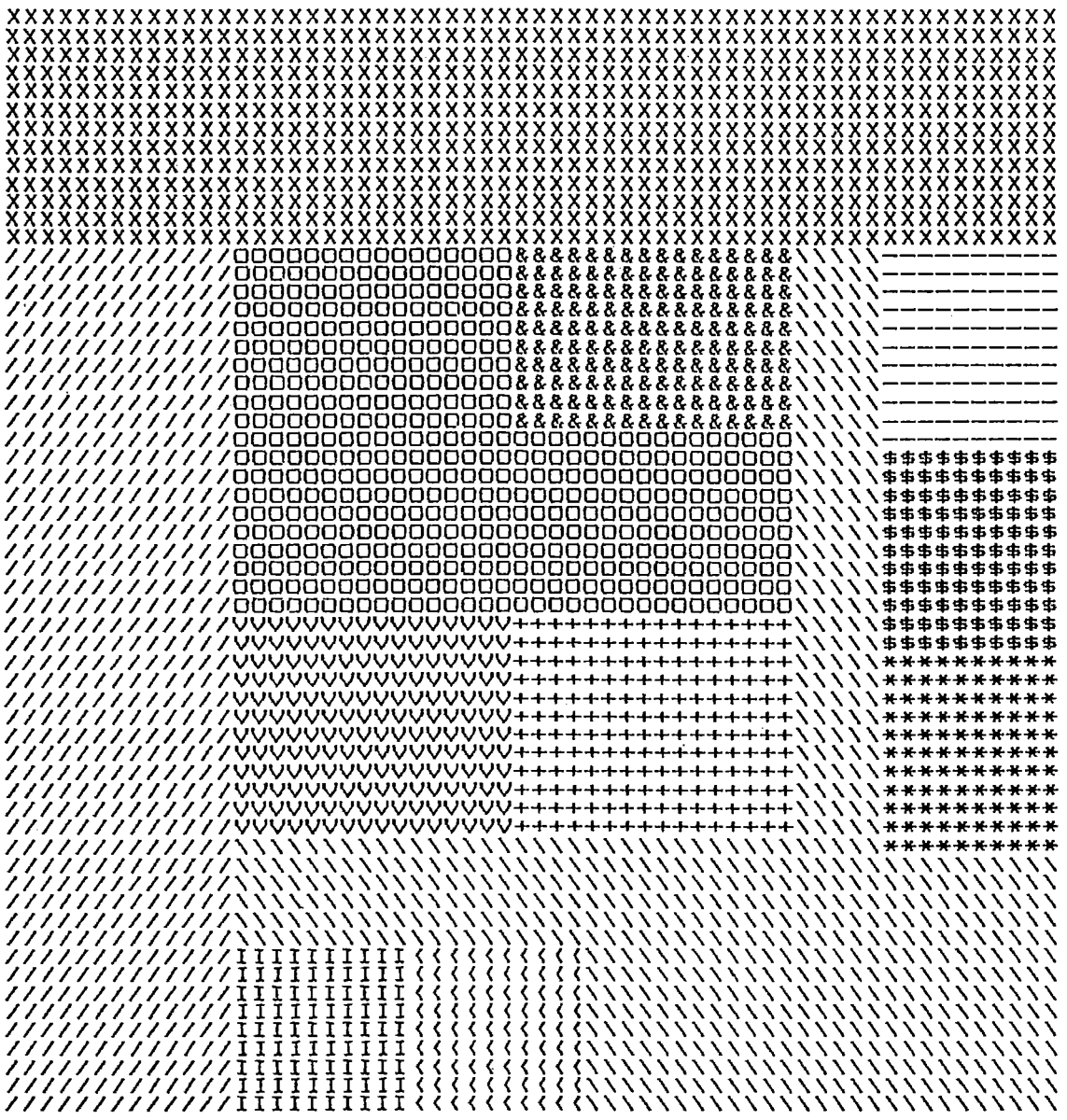
Figure 7.15 ctd.....

```
INSERTING VERTEX 10 IN TRIANGLE 4  7  5

INSERTING VERTEX 11 IN TRIANGLE 1  6  7

INSERTING VERTEX 8 IN TRIANGLE 1  7  11

INSERTING VERTEX 9 IN TRIANGLE 1  7  8

INSERTING VERTEX 13 IN TRIANGLE 6  7  4

INSERTING VERTEX 2 IN TRIANGLE 1  7  12

INSERTING VERTEX 3 IN TRIANGLE 1  7  2

TOTAL DELTAHEDRON ADJACENCY SCORE IS 509

INCIDENCE MATRIX

 0  1  1  0  0  1  1  1  1  0  1  1  0
 1  0  1  0  0  0  1  0  0  0  0  1  0
 1  1  0  0  0  0  1  0  0  0  0  0  0
 0  0  0  0  1  1  1  0  0  1  0  1  1
 0  0  0  1  0  0  1  0  0  1  0  1  0
 1  0  0  1  0  0  1  0  0  0  1  1  1
 1  1  1  1  1  1  0  1  1  1  1  1  1
 1  0  0  0  0  0  1  0  1  0  1  0  0
 1  0  0  0  0  0  1  1  0  0  0  0  0
 0  0  0  1  1  0  1  0  0  0  0  0  0
 1  0  0  0  0  1  1  1  0  0  0  0  0
 1  1  0  1  1  1  1  0  0  0  0  0  0
 0  0  0  1  0  1  1  0  0  0  0  0  0


BEST POSSIBLE SOLUTION BOUND IS 552

DELTAHEDRON SOLUTION RATIO IS  .922102

DISTANCE CLASSES:

 6  7  12  11  8  9  2  3
 4  5  10  13
```

RATIO OF BUILDING LENGTH TO WIDTH (>  = 1)? 1.0

HORIZONTAL SIZE OF PLAN MATRIX DESIRED?  60

Figure 7.15 ctd.....



CHARACTER SET CODES:

FACILITY 2 IS REPRESENTED BY I
FACILITY 3 IS REPRESENTED BY (
FACILITY 4 IS REPRESENTED BY O
FACILITY 5 IS REPRESENTED BY V
FACILITY 6 IS REPRESENTED BY X
FACILITY 7 IS REPRESENTED BY \
FACILITY 8 IS REPRESENTED BY $
FACILITY 9 IS REPRESENTED BY *
FACILITY 10 IS REPRESENTED BY +
FACILITY 11 IS REPRESENTED BY -
FACILITY 12 IS REPRESENTED BY /
FACILITY 13 IS REPRESENTED BY &

Now that specific areas are included, class cardinality is no longer a criterion for the relative sizes of the two distance classes. Let A(x) = area of facility x. With reference to Figure 7,12 define

$$SUMTL = \sum_{x \in BTL} A(x)$$

$$SUMTR = \sum_{x \in BTR} A(x)$$

$$SUMLR = \sum_{x \in BLR} A(x)$$

$$SUMI = \sum_{x \in DC_2} A(x)$$

Refer now to Figure 7.16, and assume SUMTR, SUMLR, SUMI $\neq$ 0.

Given A(TOP) and the block plan width, W, we may calculate the value of P1Y, assuming integral division. Then the value of P1X may be obtained, using the known values P1Y, RATIO*W, SUMTL and A(LEFT). If we now assume a distance of 1 'unit' between chains BTR and $DC_2$, and BLR and $DC_2$, and take the lengths x and y to be proportional to the chain areas

$$\text{i.e.:} \quad \frac{SUMTR}{x} = \frac{SUMLR}{y}$$

then the required value of $\gamma$ is found as the solution of

$$\gamma = [-\alpha + (\alpha^2 + 4\beta)^{\frac{1}{2}}]/2 \ ,$$

where $\alpha$ = SUMLR*(W-P1X)/SUMTR - RATIO*W - P1Y

and $\beta$ = SUMI*SIMLR/SUMTR.

For the special cases where either SUMTR = 0 or SUMLR = 0, $\gamma$ may be calculated as

$$\gamma = [SUMI/(SUMI + SUMTR + SUMLR + A(RIGHT))^{\frac{1}{2}} * (RATIO*W - P1Y)$$

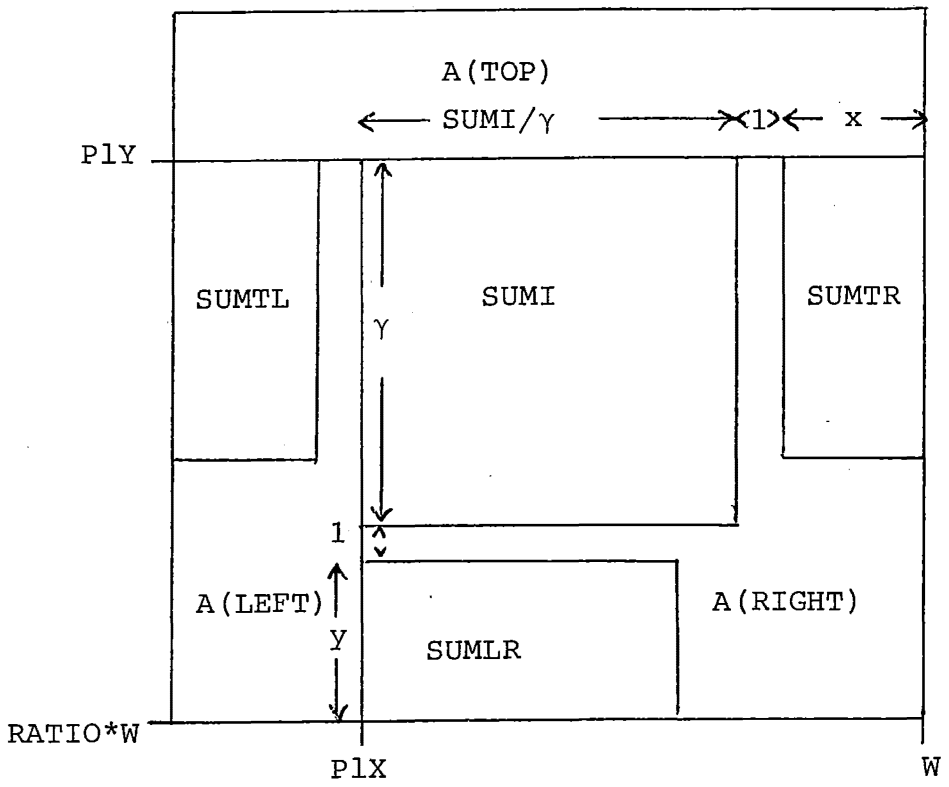



Figure 7.16 Determining the distance class delimiters

Of course, if SUMI = 0, then $\gamma = 0$. Such a result is undesirable, however, since the resultant block plan would consist of sets of nested L-shaped facilities all adjacent to the exterior. Fortunately, for problems involving more than 10 facilities, and non-pathological transportation matrices, this situation has yet to be observed.

The definition of only a single unit distance between $DC_2$ and the chains from RIGHT is made for convenience only; experiments have shown that integral division can cause inconsistencies and infeasibilities in facility boundary delineations if this distance is not specified as some positive value. As a unit width in the plan framework,

these facility 'appendages' may form the basis for corridors in a final ornamented plan.

In physical terms this also implies that some members of the chain of facilities of the type in Figure 7.13 will have corridor-accessibility to the 'parent' facility rather than direct adjacency. For practical planning this may be more desirable (although giving a layout of inferior quality in theoretical terms) than maintaining the artificial L-shaped nesting.

Realisation of the chain configurations is straightforward in the case analogous to Figure 7.13(a). Each facility area is "filled-in" until the requisite number of scaled units is exhausted. In the alternative case, the single corridor provision mentioned above is used to ensure that all required adjacencies are met correctly, while also maintaining areal proportion. This restriction is not always necessary, especially in the case of chains of length at most three. The programmed procedure endeavours to do the "best possible" given the relative areas of the members of the chain, but often it is more desirable to allow the corridor to be produced as this then creates a more compact facility.

Figure 7.17 is an example of the execution of the constrained SUPERDELTAHEDRON method operating on a randomly generated 15*15 transportation cost matrix and randomly generated areas. The perturbations required to achieve the solution are shown. The resultant block plan (Figure 7.18) displays many of the typical features, for example:

(i) facility 14 ('=') is provided with two corridors so as to allow a more compact representation.

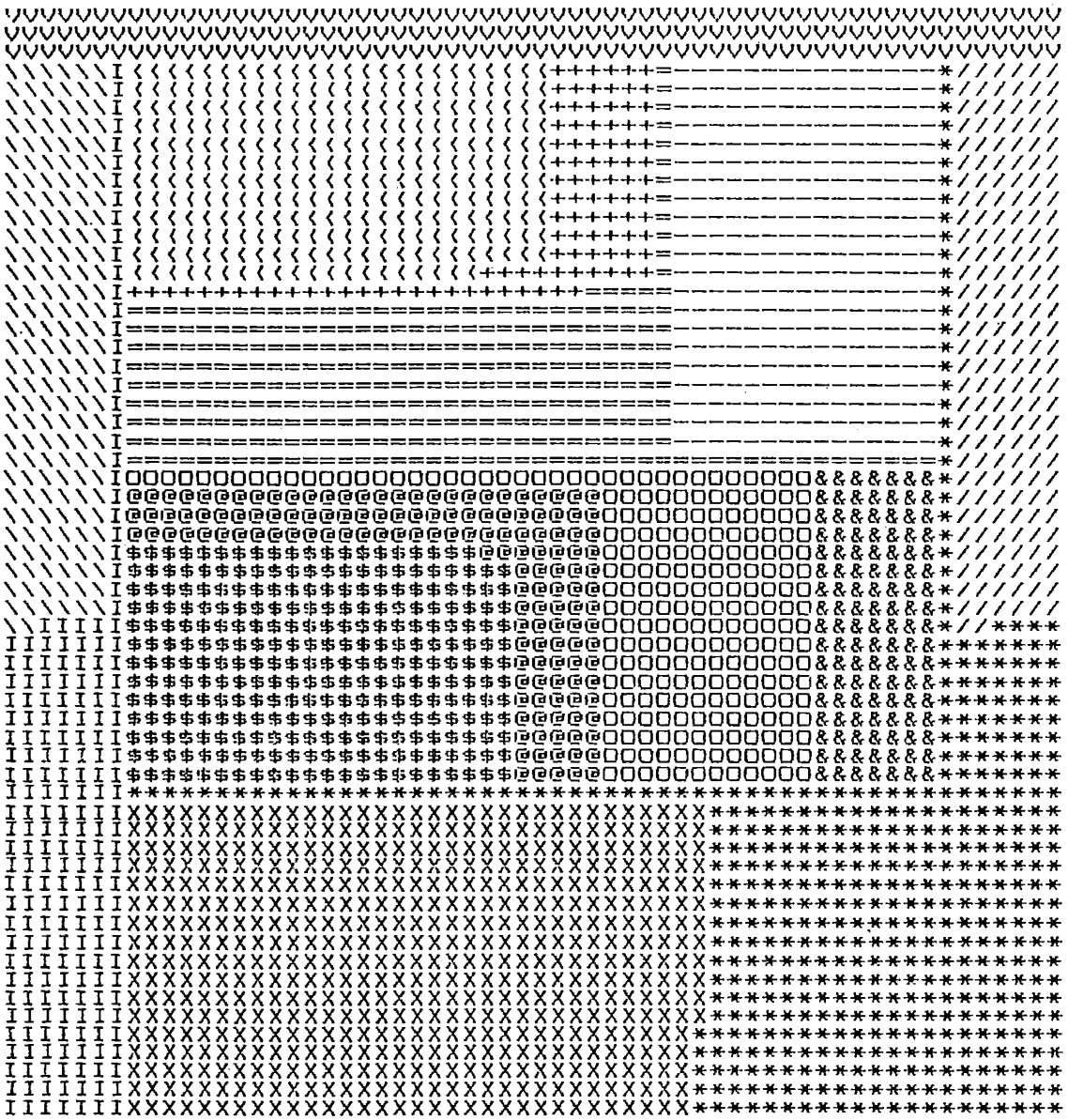
Figure 7.17

```
NUMBER OF FACILITIES: 15
GENERATING TRANSPORTATION COST MATRIX RANDOMLY

VALUE OF MU: 20
VALUE OF SIGMA: 20
TRANSPORTATION COST MATRIX:

 0  58  12  39  50  36  0  11  43  38  35  7  0  34  15
 58  0  24  0  17  0  10  0  44  15  0  14  47  20  18
 12  24  0  20  38  11  30  6  16  8  27  17  4  30  0
 39  0  20  0  39  20  0  6  21  24  40  13  13  16  37
 50  17  38  39  0  10  23  4  21  21  27  42  34  0  24
 36  0  11  20  10  0  11  0  33  4  56  35  67  17  14
 0  10  30  0  23  11  0  35  40  39  29  55  0  25  9
 11  0  6  6  4  0  35  0  24  0  10  24  14  0  25
 43  44  16  21  21  33  40  24  0  0  0  21  23  19  0
 38  15  8  24  21  4  39  0  0  0  14  48  23  46  2
 35  0  27  40  27  56  29  10  0  14  0  20  1  10  0
 7  14  17  13  42  35  55  24  21  48  20  0  20  22  26
 0  47  4  13  34  67  0  14  23  23  1  20  0  17  23
 34  20  30  16  0  17  25  0  19  46  10  22  17  0  36
 15  18  0  37  24  14  9  25  0  2  0  26  23  36  0

FACILITY AREAS (2-N) :

 100  100  64  64  196  64  100  144  36  144  64  64  64  64

BEST TETRAHEDRON:

 1  2  5  9

INSERTION ORDER :

 12  6  7  14  4  13  10  11  3  15  8

INSERTING VERTEX 12 IN TRIANGLE  1  5  9

INSERTING VERTEX 6 IN TRIANGLE  1  2  9

INSERTING VERTEX 7 IN TRIANGLE  1  2  5

INSERTING VERTEX 14 IN TRIANGLE  2  5  9

INSERTING VERTEX 4 IN TRIANGLE  2  9  14


ORIENTED CYCLE:

 2 : 14  13  4
```

Figure 7.17 ctd....

```
 2 : 14  13  4

INSERTING VERTEX 13 IN TRIANGLE  4  9  14

** PERTURBING DEGREE CONSTRAINT ON VERTEX 2  **

** PERTURBING DEGREE CONSTRAINT ON VERTEX 5  **

** PERTURBING DEGREE CONSTRAINT ON VERTEX 9  **

INSERTING VERTEX 10 IN TRIANGLE  2  5  14

** PERTURBING DEGREE CONSTRAINT ON VERTEX 2  **

** PERTURBING DEGREE CONSTRAINT ON VERTEX 5  **

** PERTURBING DEGREE CONSTRAINT ON VERTEX 14  **

INSERTING VERTEX 11 IN TRIANGLE  14  5  9

** PERTURBING DEGREE CONSTRAINT ON VERTEX 5  **

** PERTURBING DEGREE CONSTRAINT ON VERTEX 9  **

** PERTURBING DEGREE CONSTRAINT ON VERTEX 14  **

INSERTING VERTEX 3 IN TRIANGLE  2  5  10

** PERTURBING DEGREE CONSTRAINT ON VERTEX 2  **

** PERTURBING DEGREE CONSTRAINT ON VERTEX 5  **

INSERTING VERTEX 15 IN TRIANGLE  2  9  4

** PERTURBING DEGREE CONSTRAINT ON VERTEX 2  **

** PERTURBING DEGREE CONSTRAINT ON VERTEX 9  **

INSERTING VERTEX 8 IN TRIANGLE  2  9  15

SUPER_DELTAHEDRON SOLUTION SCORE :  28617

DEGREE CONSTRAINTS (2..N) :

 12  8  7  11  10  7  8  12  6  9  7  7  9  7

DEGREES ACHIEVED (2..N) :

 11  3  5  9  3  3  3  11  4  3  3  3  7  4

SHORTEST PATH MATRIX VIA SUPER_DELTAHEDRON:

 0  5  13  14  4  7  4  15  6  11  14  4  16  12  14
 5  0  10  9  9  12  9  10  11  8  19  9  17  9  9
 13  10  0  19  9  20  17  20  19  8  19  17  23  15  19
 14  9  19  0  16  21  18  17  10  15  18  18  8  8  8
 4  9  9  16  0  11  8  19  10  7  10  8  16  8  18
 7  12  20  21  11  0  11  22  13  18  21  11  23  19  21
 4  9  17  18  8  11  0  19  10  15  18  8  20  16  18
 15  10  20  17  19  22  19  0  11  18  23  19  21  19  9
 6  11  19  10  10  13  10  11  0  17  12  10  10  10  10
 11  8  8  15  7  18  15  18  17  0  17  15  15  7  17
 14  19  19  18  10  21  18  23  12  17  0  18  18  10  22
 4  9  17  18  8  11  8  19  10  15  18  0  20  16  18
 16  17  23  8  16  23  20  21  10  15  18  20  0  8  16
 12  9  15  8  8  19  16  19  10  7  10  16  8  0  16
 14  9  19  8  18  21  18  9  10  17  22  18  16  16  0

DISTANCE CLASSES:

 2  5  9  12  6  7
 14  4  13  10  11  3  15  8
```

Figure 7.18



CHARACTER SET CODES:

FACILITY 2 IS REPRESENTED BY I
FACILITY 3 IS REPRESENTED BY (
FACILITY 4 IS REPRESENTED BY O
FACILITY 5 IS REPRESENTED BY V
FACILITY 6 IS REPRESENTED BY X
FACILITY 7 IS REPRESENTED BY \
FACILITY 8 IS REPRESENTED BY $
FACILITY 9 IS REPRESENTED BY *
FACILITY 10 IS REPRESENTED BY +
FACILITY 11 IS REPRESENTED BY -
FACILITY 12 IS REPRESENTED BY /
FACILITY 13 IS REPRESENTED BY &
FACILITY 14 IS REPRESENTED BY =
FACILITY 15 IS REPRESENTED BY @

(ii) facility 4 ('0') has one corridor, facility 15 ('@') retains its L-shape.

(iii) due to integral division (and the calculation of $\gamma$), the area of facility 13 is slightly smaller than required.

Although not necessary in this example, if the variation in facility areas of a problem is sufficiently large, the situation of a small facility nested between two large facilities may occur. Suitable perturbations of areas may then be needed to prevent distortion or incomplete adjacency. In program testing, area variation was kept sufficiently low so as to minimize the possibility of such occurrences.

The two programs, 'BLOCK' and 'SPLAN', which implement the described methodology are included in Appendix 2. Both are written in Microsoft BASIC version 5.0. Program 'BLOCK' allows for two forms of input: a relationship matrix (which then leads to the DELTAHEDRON method, followed by program 'PLAN') or a transportation cost matrix (which invokes SUPERDELTAHEDRON, followed by program 'SPLAN'). This effectively combines the two approaches into a compact form, allowing useful planning flexibility. Total execution time for the example problem was 350 seconds, again using the BASIC interpreter.

We now give a brief performance comparison of the present method with the codes CRAFT, CORELAP and ALDEP. The problem to be considered is based on a seven-facility example from Tompkins and Moore (1978), whose input data is summarized in Figure 7.19. Additional data in the form of transportation costs for the exterior facility, is required for our formulation.

| Facility | | Area (000 sq units) |
|---|---|---|
| A | Receiving | 12 |
| B | Milling | 8 |
| C | Press | 6 |
| D | Screw machine | 12 |
| E | Assembly | 8 |
| F | Plating | 12 |
| G | Shipping | 12 |
| | | 70 |

RELATIONSHIP CHART

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| A | | E | O | I | O | U | U |
| B | | | U | E | I | I | U |
| C | | | | U | U | O | U |
| D | | | | | I | U | U |
| E | | | | | | A | I |
| F | | | | | | | E |
| G | | | | | | | |

WEIGHTING FACTORS

| | ALDEP | CORELAP |
|---|---|---|
| A | 128 | 243 |
| E | 32 | 81 |
| I | 8 | 27 |
| O | 2 | 9 |
| U | 0 | 1 |

TRANSPORTATION COST MATRIX:

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| A | 0 | 45 | 15 | 25 | 10 | 5 | 0 |
| B | 45 | 0 | 0 | 50 | 25 | 20 | 0 |
| C | 15 | 0 | 0 | 0 | 5 | 10 | 0 |
| D | 25 | 50 | 0 | 0 | 35 | 0 | 25 |
| E | 10 | 25 | 5 | 35 | 0 | 70 | 35 |
| F | 5 | 20 | 10 | 0 | 70 | 0 | 65 |
| G | 0 | 0 | 0 | 25 | 35 | 65 | 0 |

Figure 7.19 Data for example problem

Although CRAFT uses the transportation cost data and CORELAP and ALDEP the relationship chart, each layout produced will be evaluated in terms of the rectilinear metric of SUPER_DELTAHEDRON. This allows direct comparison with the results obtained from programs "BLOCK" and "SPLAN". (Given that neither DELTAHEDRON nor "PLAN" takes into account facility areas, only the SUPER_DELTAHEDRON approach is considered.)

Figures 7.20 - 7.22 outline the results of CRAFT, CORELAP and ALDEP; Figures 23-24 give two outputs from "SPLAN", corresponding to whether or not facility A is forced to be adjacent to the exterior. Table 7.1 shows the scores associated with each layout, as generated from

| LAYOUT METHOD | SOLUTION SCORE | |
|---|---|---|
| CRAFT | 2302.8 | (2122.1) |
| CORELAP | 2177.8 | (2079.1) |
| ALDEP | 2539.0 | |
| BLOCK(1) | 1824.5 | |
| BLOCK(2) | 1768.1 | |

Table 7.1 Solution scores in terms of the SUPER_DELTAHEDRON metric

their dual graphs. It is possible to improve the CRAFT and CORELAP ratings by simple perturbations; for CRAFT we eliminate the two 4-way junctions, and introduce adjacency between facilities A and F, and B and E; for CORELAP we make facilities D and G adjacent. These modifications have the effect of maximizing the number of "within-in plan" adjacencies, without generating a maximal planar dual graph. The resultant scores are supplementary to Table 7.1.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | A | A | A | A | A | A | A | A | A | A | G | G | G | G | G | G | G | G |
| 2 | A | | | | | | | | | A | G | | | | | | | G |
| 3 | A | A | A | A | A | A | A | A | A | A | G | G | G | | | | | G |
| 4 | C | C | C | B | B | B | B | B | B | B | F | F | G | G | G | G | G | G |
| 5 | C | | C | C | B | | | | | B | F | F | F | F | F | F | F | F |
| 6 | C | | | C | B | B | B | B | B | B | F | F | F | F | F | F | | F |
| 7 | C | C | C | C | B | D | D | D | D | E | E | E | E | E | E | F | | F |
| 8 | D | D | D | D | D | D | | | D | E | | | | | E | F | | F |
| 9 | D | | | | | | | D | D | E | E | E | E | E | E | F | | F |
| 10 | D | D | D | D | D | D | D | D | H | H | H | H | H | E | E | F | F | F |

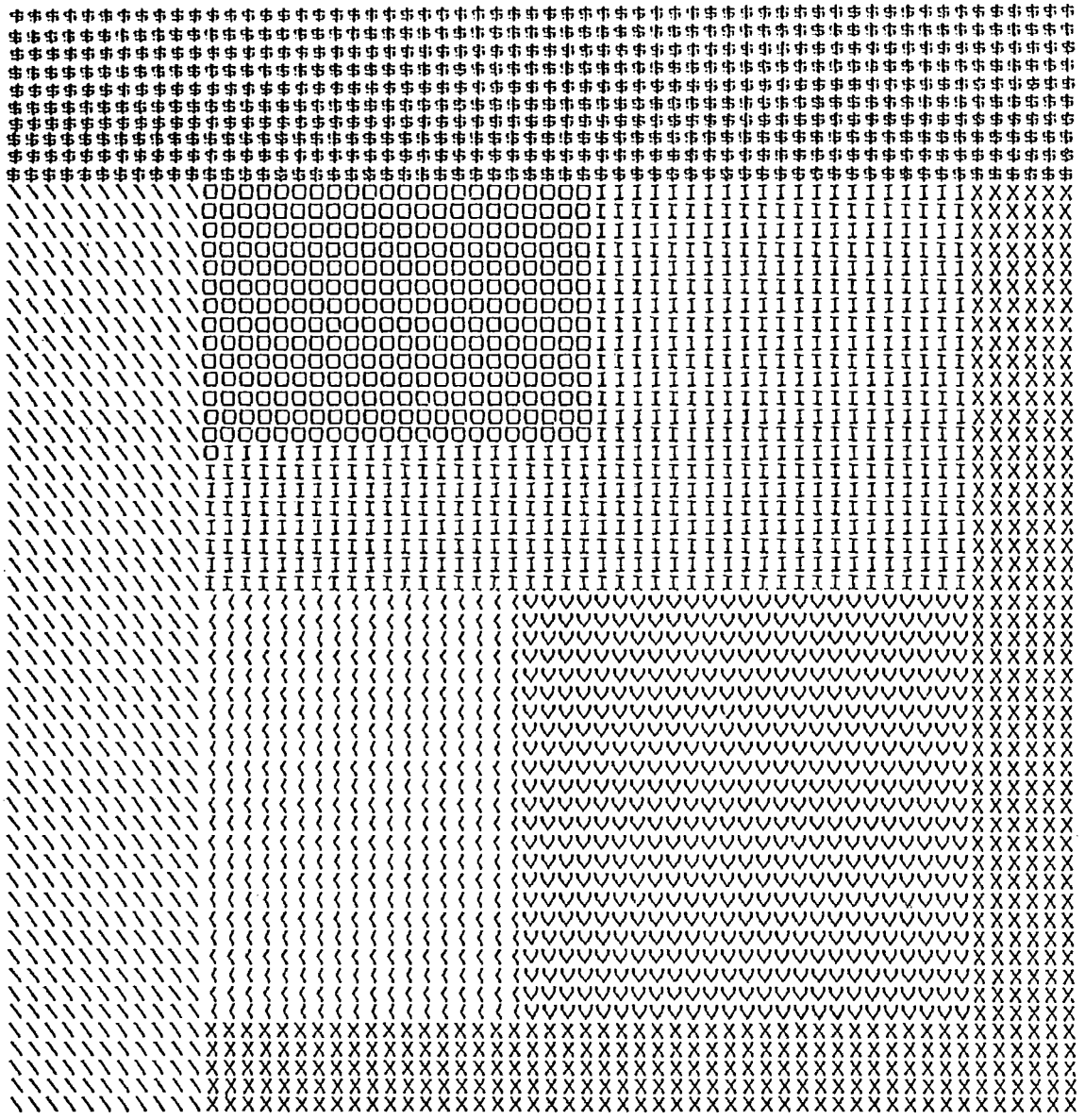
Figure 7.20 The output from CRAFT
( H is a "dummy" facility)

| | | | |
|---|---|---|---|
| D | A | A | C |
| D | B | F | F |
| | G | G | E |

Figure 7.21 The output from CORELAP

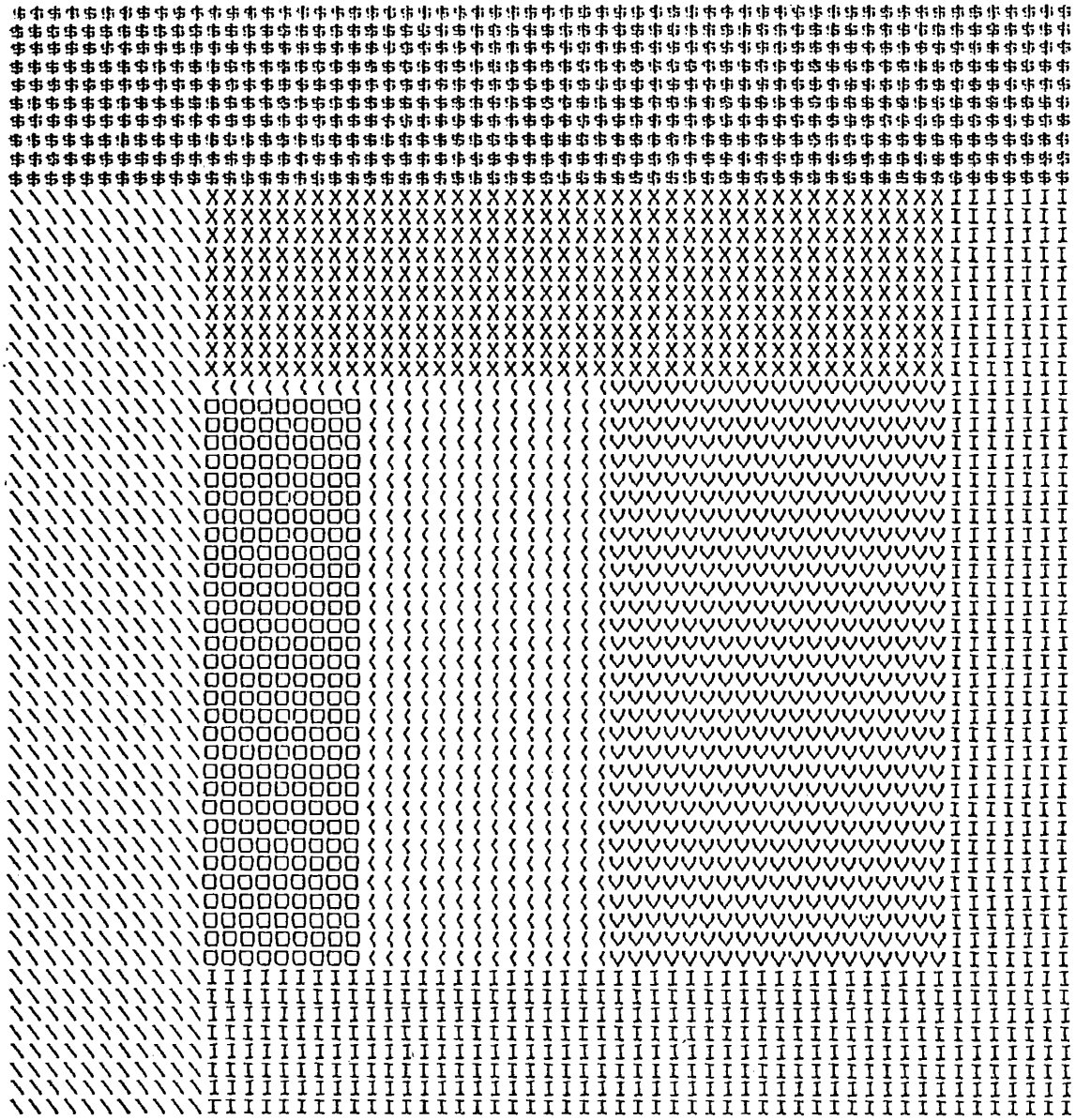
| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | A | B | B | B | B | D | D | E | E | F | F | F | F | G | G | G | G |
| A | A | B | B | B | B | D | D | E | E | F | F | F | F | G | G | G | G |
| A | A | B | B | B | B | D | D | E | E | F | F | F | F | G | G | G | G |
| A | A | B | B | B | B | D | D | E | E | F | F | F | F | G | G | G | G |
| A | A | B | B | B | B | D | D | E | E | F | F | F | F | G | G | G | G |
| A | A | A | A | D | D | D | D | E | E | F | F | C | C | G | G | G | G |
| A | A | A | A | D | D | D | D | E | E | F | F | C | C | G | G | G | G |
| A | A | A | A | D | D | D | D | E | E | F | F | C | C | C | G | | G |
| A | A | A | A | D | D | D | D | E | E | F | F | C | C | C | C | | |
| A | A | A | A | D | D | D | D | E | E | F | F | C | C | C | C | | |

Figure 7.22 The output from ALDEP



| FACILITY | CODE |
|---|---|
| A | I |
| B | < |
| C | 0 |
| D | V |
| E | X |
| F | \ |
| G | $ |

Figure 7.23 BLOCK(1) output
$t_{A,EXT} = 50$, $t_{G,EXT} = 50$
(A not adjacent to exterior)



| FACILITY | CODE |
|---|---|
| A | I |
| B | < |
| C | 0 |
| D | V |
| E | X |
| F | \ |
| G | $ |

Figure 7.24 Output BLOCK(2)
$t_{A,EXT} = 106$, $t_{G,EXT} = 90$
(A adjacent to exterior)

It is instructive to calculate how accurately the solution scores are reflected in the corresponding layout. Under the assumption of complete pairwise rectilinear communication between facility centroids, we can determine and estimate of the actual distance travelled. We note that, in general, such communication is not available - travel is usually via corridors or paths between facility boundaries, as generated, for example, by the method of Gawad and Whitehead (1976). This approach is an augmentation step, building on the initial layout. Table 7.2 summarizes the results of the calculations.

| METHOD* | RECTILINEAR SCORE |
|---|---|
| CRAFT (perturbed) | 2139.1 |
| CORELAP (perturbed) | 2200** |
| BLOCK(1) | 3002.4 |
| BLOCK(2) | 2683.7 |

Table 7.2 Rectilinear scores of the layouts

* ALDEP not included - output scale too large

** estimate

Comparison of Tables 7.1 and 7.2 shows that the SUPER_DELTAHEDRON metric has severely underestimated the cost of the layouts obtained via "BLOCK" and "SPLAN". In this case it is clearly because of the elongated nature of the facilities adjacent to the exterior. These distort the SUPER_DELTAHEDRON assumption of each facility being a square. Consider, for example, the triad E,F,G; in BLOCK(1) their mutual transportation costs account for 36.2% of the

total layout cost according to the SUPER_DELTAHEDRON criterion, whereas for the actual rectilinear evaluation the comparable figure is 59.1%. The corresponding figures for BLOCK(2) are 28.1% and 42.6%, while for CRAFT they are 40.8% and 36.8%. Hence, strict adherence to the maximal planar graph structure, characterised by an exterior ring of three facilities (e.g. E,F,G in BLOCK(1)) can clearly result in an inferior solution, especially in an example with few facilities, where the relative distortion will be higher. Simple perturbations of the solution are possible, here, however. Figure 7.25 displays one possible modification of BLOCK(2), after noting that $t_{28} = 0$; its SUPER_DELTAHEDRON score is 1796.3, but its rectilinear equivalent is reduced to 2166.7, a score comparable with both the perturbed CRAFT and CORELAP solutions. Note that the corresponding dual graph is not maximally planar.

The preceding discussion has pointed out potential difficulties in the use of the SUPER_DELTAHEDRON metric. The example cited is a special case - those facilities with a higher level of distortion also happen to have the greatest degree of intercommunication. In larger problems this is less likely to occur, despite retention of the same basic layout structure. Thus, while some refinements may still be necessary to the rectilinear score estimation and the method, it has at least provided an initial systematic graph theoretic approach to the problem of facilities layout, potentially capable of producing layouts of a quality similar to that of existing methods for problems involving up to 20 facilities.

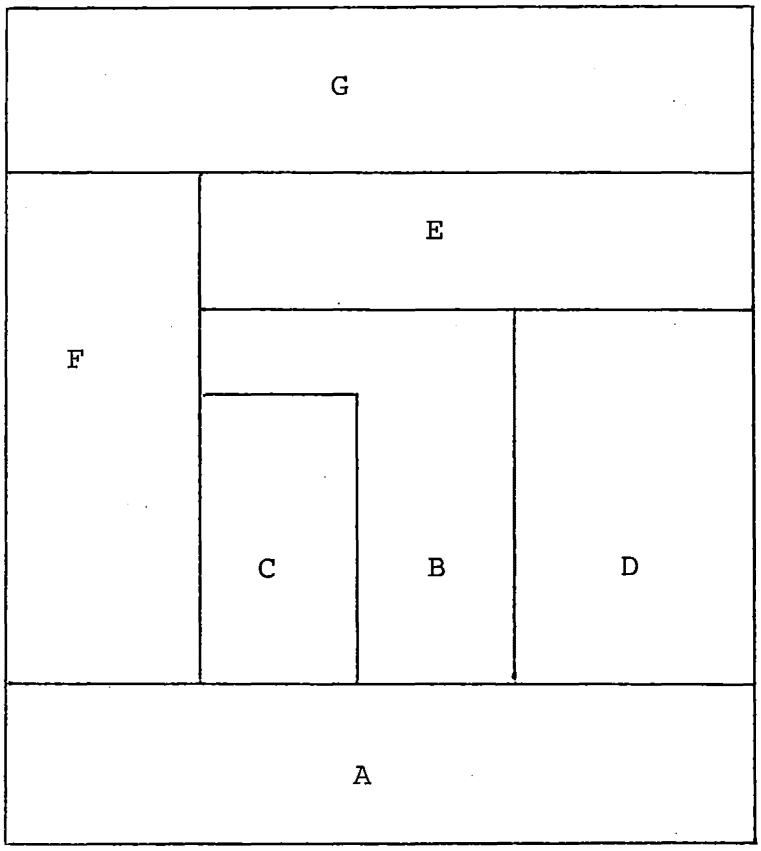



Figure 7.25 A possible perturbation of the BLOCK(2) solution

# CHAPTER 8

# MULTI-FLOOR BUILDING LAYOUT

## 8.1 Survey of previous work

The work of Archer (1963) appears to be the first attempt at modelling multi-floor layout. Facilities are allocated positions on the basis of their relationship and distance from the most "important" facility, with no distinction being made between horizontal and vertical travel.

Mosely (1963) subsequently improved and corrected Archer's approach in a modified Transportation Problem format, minimizing cost in terms of the total distance travelled per time period. Vertical circulation zones and entrances are prelocated in the plan. However, the analysis concentrates mainly on circulation problems and determining the minimum cost building shape rather than the actual layout of facilities. The linearisation also implicitly assumes that all journeys commence from a common point of origin.

The computer program ALDEP (Seehof et. al (1967)) was developed to handle buildings with at most three floors with the objective of maximizing adjacency scores. Layouts are generated partly constructively and partly by improvement strategies, using random location techniques. Required input includes the Relationship matrix, facility areas, facility prelocation information, and space availability constraints.

Vertical circulation is not defined as an activity; only within-floor associations are considered.

Willoughby (1970), (1971) also employs a constructional technique. So-called "absolute association groups" are identified within the relationship matrix, and floor schedules created corresponding to the given floor areas. Vertical and horizontal circulation routes are provided manually, and each facility is assigned a subjective relationship with these. Distances are measured diagonally, from centroid to centroid, and a process of "sequential grid summation and updating" is used to select final placement positions for activities on a grid for each floor. Facility shapes may be defined, and the location ordering may be altered or restarted if an impasse is encountered. Again, however, there is no cost relationship defined between vertical and horizontal movement. The level of intuitive assessment required is also large.

Portlock and Whitehead (1974) considered three construction-type methods. Their "Unlimited Access" approach assumes that vertical movement is possible at every point within the plan boundaries. Facilities are decomposed into unit elements, and element location is on the basis of minimum cost with respect to these elements already placed. This cost, C(i), for the $i^{th}$ element is given by

$$C(i) = \sum_{k=1}^{i-1} [r_{ik}(d_{ik} + v|F_i - F_k|)]$$

where r is the elemental relationship

d is horizontal distance

v is the vertical distance factor

F represents the floor number

When all elements have been placed, the best position for the vertical circulation zone is then determined.

The "Floor Listing" approach breaks down the problem into a set of two-dimensional problems. It is assumed that figures representing the relative importance of positioning each facility on each floor is available; the relationship scores and this floor desirability data are used to create autonomous facility clusters with minimal interconnections, with the proviso that any facility must be wholly located on a single floor. Vertical circulation zones are then introduced, and the relationship of each vertical access point with facilities on the same floor is made proportional to the sum of the relationships between each facility and all other facilities not on that particular floor.

The "Movement Simulation" approach attempts an explicit definition of the floor-linking function. Vertical circulation zone positions are pre-specified. Each level in the layout is considered in turn, and each unoccupied grid is costed as follows:

for elements i and the k other elements on the same floor, costs are assigned via

$$C(i) = \sum_{j=1}^{k} r_{ij} d_{ij}$$

where, for an element pair (i,j) on different floors

$$d_{ij} = \underset{k}{\text{MIN}}\{d_{ik} + v|F_i - F_j| + d_{kj}\}$$

for travel via each circulation zone k.

Location at positions of minimum cost is maintained throughout the construction. This approach is the most realistic of the three, but may require several complete iterations for testing various circulation zone placements.

Carter and Whitehead (1975) modified some of Portlock and Whitehead's ideas. "Natural" facility groups are identified from the relationship matrix using cluster analysis. Areas are then introduced, resulting in a breaking-down of the group of clusters into partitions of appropriate area (for each floor). Floor partitions are then arranged into a minimum cost vertical layout by considering inter-floor associations and the existence of any prelocated facilities. The layout of each floor uses the two-dimensional construction technique of the "Floor Listing" approach.

Floor areas are buffered by 5% to accommodate many possible configurations. Only 1 vertical circulation zone, located in a central position, is allowed per floor, avoiding unnecessary increases in the "total distance function" of a layout resulting from eccentric position of the zones (Tabor (1970)). The authors note that prelocated facilities can increase movement costs by up to 10%; care must be taken to ensure that such facilities are in close proximity to their "natural" clusters.

Chyutin and Mittwoch (1979) consider a different formulation, that of optimal ingress to and egress from a high-rise building. i.e.: allocation facilities to positions so as to minimize the total vertical traffic within a building for a given total of anticipated traffic to each facility from outside the building and vice versa.

White (1972) describes a method for three-dimensional hospital layout, as a linearisation of the Quadratic Assignment Problem. The objective is the minimization of total travel cost between all facility pairs, where the travel costs are in terms of "trip production" coefficients, $t_{ij}$,

Johnson chose to minimize the total variables cost of movement between facilities

$$\text{i.e.:} \quad \text{MIN} \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} v_{ij}$$

where $t_{ij}$ = time for travel from facility i to facility j

$v_{ij}$ = number of journeys from i to j, scaled as required.

$v_{ij}$ is assumed constant for a particular problem; $t_{ij}$ is assumed proportional to the rectilinear distance between the centroids of facilities i and j, including the non-linear vertical component (defined as via the 'fastest' vertical circulation point).

Floors are divided into types with all floors of the same type having identical shape and layout. Each floor is divided into an integral number of congruent rectangular modules and each facility is represented by an integral number of such modules. If possible, all facilities must be located contiguously. In the case where facilities must be split between floors, the volume of movement from and to each subfacility is assumed proportional to the fraction of modules of the whole facility that each subfacility contains.

Sub-ground-floor levels are permitted, as are elevators running only between specified floors and prelocated facilities other than the elevators.

The improvement procedure of SPACECRAFT is similar to that of CRAFT. From the initial layout, a better solution is iteratively created by exchanging the two or three departments which yield the greatest nett savings. Candidate triples for exchange must be contiguous, and non-adjacent pairs of equal size, and the resultant layout must not violate any

inputted restrictions on facility location or shape. The procedure terminates when no further improvements can be found.

As the heuristic uses relative costs in an ordinal manner, it was found to be very robust to (non-linear) cost reformulations. Unfortunately, the allowance of prelocated facilities meant that the number of exchange alternatives was often significantly reduced. Perturbations in facility sizes were also occasionally required to generate more equally-sized facilities, leading to greater flexibility in exchanges.

Johnson noted that poor layouts would result in a higher utilization of elevators, so that vertical travel times should be updated to reflect improved layouts, rather than assuming times for a good layout a priori. This modification would, of course, significantly increase the complexity of the procedure, which, within the limitations of the improvement rationale, appears to be a successful extension of CRAFT.

## 8.2 A Graph Theoretic Model for Multi-Floor Layout

The methods described in the previous section cover all the essential ingredients of multifloor layout via three categories

(i) construction [e.g.: Portlock and Whitehead]

(ii) improvement [e.g.: SPACECRAFT]

(iii) agglomeration [e.g.: Carter and Whitehead]

In the graph theoretic framework we may encompass all three categories in the following approach:

(a) Partition the facilities (excluding the exterior facility) into groups whose areas correspond to the building floor areas; inter-group "communication" should be minimized

(b) Assign the groups to floors in a cost-minimizing configuration

(c) Obtain, constructively, a high-quality adjacency graph for each floor, relative to a vertical circulation zone using

(i) the DELTAHEDRON heuristic, or

(ii) the SUPER_DELTAHEDRON heuristic;

improve this graph until a local optimum is obtained.

(d) Develop the corresponding block plan from the acjacency graph for each floor, ensuring consistent elevator placement and reintroducing the exterior facility, for either c(i) or c(ii).

Our formulation of the Multi-floor layout problem (MULTI_FLOOR) in the SUPER_DELTAHEDRON framework is as follows:

Let N = number of facilities

$f_i$ = representation of floor i, i=1,...,M

$a_i$ = area of facility i, i=1,...,N

$A_k$ = area of floor $f_k$, k=1,...,M

h = highest common factor of $a_i$, i=1,...,N

$p_i$ = number of unit modules in facility i

= $a_i$ div h, i=1,...,N

$m_i$ = set of modules representing facility i

= $\{m_{ij}\}$, i=1,...,N, j=1,...,$p_i$

$A_k^*$ = number of unit modules in floor $f_k$

$G_k = (V_k, E_k)$ = the adjacency graph representing floor $f_k$

$$\mu_{ijk} = \begin{cases} 1 \text{ if } m_{ij} \in V_k, & i=1,\ldots,N \\ & j=1,\ldots,p_i \\ & k=1,\ldots,M \\ 0 \text{ otherwise} \end{cases}$$

$e_i$ = elevator facility for floor $f_i$ ($e_i \in v_i$)

$t_{ij}$ = time taken to travel from module i to module j (see later for the definition of $t_{ij}$)

$= t_{ji}$

$v_{ij}$ = number of journeys between facilities i and j per time period

$= v_{ji}$

For modules our definition becomes:

$$v_{ij} = \begin{cases} L \text{ if modules } i,j \in \text{facility } k \text{ (L large)} \\ \\ v_{kr} \text{ if module } i \in \text{facility } k, \text{ module } j \in \text{facility } r \ (k \neq r) \end{cases}$$

We also assume that the building area available and the sum of facility areas required are concomitant,

$$\text{i.e.:} \quad \sum_{i=1}^{N} p_i = \sum_{k=1}^{M} A_k^*$$

Then problem MULTI_FLOOR takes the following form:

$$\text{MINIMIZE} \quad \sum_{i=1}^{N} \sum_{j=1}^{N} t_{ij} v_{ij}$$

subject to:

$$\sum_{i=1}^{N} \sum_{j=1}^{p_i} \mu_{ijk} = A_k^*, \qquad k=1,\ldots,M$$

$$\sum_{k=1}^{M} \mu_{ijk} = 1 \qquad i=1,\ldots,N \quad j=1,\ldots,p_i$$

$$\mu_{ijk} = 0/1$$

$$G_k = (V_k, E_k) \text{ maximal planar}$$

Because of the embedded maximal planar graph sub-problems (one for each floor of the layout), this problem is NP-complete, necessitating a heuristic approach. The form of approach outlined is really the only viable graph theoretic alternative because of the constructive nature of the method; pairwise distances between all facilities may only be defined once all the edges of the maximal planar graphs representing each floor have been defined. This interdependence is partially overcome by an analogy with the approach of Carter and Whitehead... treating the layout of each floor independently of the others, except that proximity of a facility to the elevator facility is governed by its level of communication with other floors.

The formulation of the problem in terms of the DELTAHEDRON rationale is essentially similar, except that the objective becomes the maximization of intra-floor adjacency scores (corresponding to the minimization of inter-floor adjacency scores. Initial stages of both problems may therefore be thought of as requiring a vertex partition with minimal edge-weight interconnection. Throughout the

remainder of this chapter we will deal primarily with the SUPER_DELTAHEDRON approach of minimizing transportation costs; reference will be made to the parallel requirements of DELTAHEDRON, if differences exist.

An alternative approach to partitioning would be to require that floor preference ratings be defined for each facility; these ratings may take two forms:

(i) for each $v \in V$, define

$PR^1(v) = k$ <=> it is desired that facility v be positioned on floor k

(ii) for each $v \in V$, $1 \leq k \leq M$, define

$PR^2(v,k) = r$ <=> the value of assigning facility v to floor k is r, $(r \in Z^+)$.

If we assume that

$$PR^2(v,k_i) > PR^2(v,k_j)$$

implies a greater desirability, then for (ii) we would want to maximize the overall facility-floor assigning in terms of these $PR^2$ values; for (i) we would seek a facility-floor assignment which satisfies a maximal number of $PR^1$ requirements. However, the difficulty of predefining either the $PR^1$ or $PR^2$ values with sufficient accuracy, reliability and consistency, together with the inherent bias involved leads us to reject this approach in favour of the interconnection weight method.

We now describe in detail the steps involved in (a) and (b), and outline a proposed method of approach for (d). Part (c) has, of course, been covered in Chapters 3 and 6.

STEP (a): Determination of the minimal cost partition

The problem of determining the minimal cost partition of the vertices into groups with minimal edge-weight inter-

connection may be defined as follows:

Consider a graph G = (V,E) with vertex-weights $r(v) \in Z^+ \; \forall \; v \in V$ (in fact, $r(v) = p_v$ from above) and edge-weights $w(e) \in Z^+ \; \forall \; e \in E$. Then we require a partition of V into disjoint equally-sized sets $V_1, V_2, \ldots, V_m$ such that

$$\sum_{v \in V_i} r(v) = K \quad , \quad 1 \leq i \leq M$$

under

(CONDITION 1) $\sum_{e \in E'} w(e)$ is MINIMIZED,

where $E' \subseteq E$ is the set of edges of E having their end vertices in different sets.

If we replace (CONDITION 1) by (CONDITION 2)

$$\sum_{e \in E'} w(e) \leq J \quad (\; J > 0, \text{ given})$$

and request the <u>existence</u> of the desired partition, then this problem is NP-complete (Hyafil and Rivest, 1973). Notice that our original problem may be couched in terms of the modified problem by the process:

(i) choose an initial value of $J > 0$

(ii) if a partition exists with CONDITION 2 then go to (iii) else go to (iv)

(iii) <u>repeat</u>

$J = J - 1$

<u>until</u> no partition exists with CONDITION 2;

identify the last successful partition; STOP

(iv) <u>repeat</u>

$J = J + 1$

<u>until</u> a partition exists with CONDITION 2;

choose this successful partition; STOP.

Hence, the NP-complete decision problem (encompassing CONDITION 2) is merely a restriction of our partitioning problem; thus, the partitioning problem is also NP-complete. So again we must turn to a heuristic method, in this case that of Kernighan and Lin (1970), which, for the sake of completeness, we now briefly describe. The approach is applicable to both DELTAHEDRON and SUPER_DELTAHEDRON.

Firstly, consider partitioning the vertices of a graph (G,w) into only two sets. Suppose we have an arbitrary partition $P = [Q|R]$, $|Q| = |R| = n$

Let $e(q) = \sum_{x \in R} w_{qx}$ $\quad \forall\, q \in Q$

be the external cost of vertex q

and $i(q) = \sum_{y \in Q} w_{qy}$ $\quad \forall\, q \in Q$

be the internal cost of vertex q.

Similarly for the elements of R:

$e(r) = \sum_{x \in Q} w_{rx}$ $\quad \forall\, r \in R$

and $i(r) = \sum_{y \in R} w_{ry}$ $\quad \forall\, r \in R$

Let $d(p) = e(p) - i(p)$ $\quad \forall\, p \in P$

Lemma: (Kernighan and Lin)

Let $q \in Q$ and $r \in R$. Then if q and r are interchanged between the sets, then the resultant change in cost is

$$d(q) + d(r) - 2w_{qr} .$$

The "two-way" partitioning procedure takes the form of Figure 8.1.

```
PROCEDURE TWO_WAY_PARTITION(Q,R);
repeat
    Calculate d(p) ∀ p ∈ P = [Q|R];
    k := 1;
    for k := 1 to n do
    begin
        choose q ∈ Q, r ∈ R such that
        g_k := d(q) + d(r) - 2w_qr gives greatest cost reduc-
               tion;
        q_k := q; r_k := r;
        Q := Q - q; R = R - r;
        d(x) = d(x) + 2w_xq_k - 2w_xr_k ∀ x ∈ Q;
        d(y) = d(y) + 2w_yb_j - 2w_yq_k ∀ y ∈ R;
    end;
    choose s to maximize G = Σ_{i=1}^{s} g_i;
    Q := Q - {q_1,...,q_s} + {r_1,...,r_s};
    R := R - {r_1,...,r_s} + {q_1,...,q_s};
until G = 0;
```

Figure 8.1 Two-way partitioning

TWO_WAY_PARTITION has complexity $0(n^2 \log n)$. This routine may easily be extended to unequal sized sets Q* and R* by restricting the maximum number of exchange candidates to MIN(|Q*|,|R*|) at each pass.

In order to determine the initial partition of V into $V_1, V_2, \ldots, V_m$, the present implementation uses two techniques:

(i) sequential break-off

(ii) random division.

Sequential break-off, as the name suggests, initially partitions a set of $k_n$ elements into two sets of n and (k-1)n

elements, using the extension of procedure TWO_WAY_PARTITION to unequal sized sets. The set of (k-1)n elements is then further broken into sets of n and (k-2)n elements. This process continues until the k required sets are obtained. An "error" in the identification of the first set broken off may lead to bias in the creation of further sets; if the initial $k_n$ element set is ordered lexicographically rather than in natural clusters, little change in the order appears to occur in the subsequent pair-wise optimization.

The use of random partitioning can lead to a drastically different initial partition that proves useful for comparison purposes. The method used is based on the scheme shown in Figure 8.2 for generating a random k-element subset R of a set $\{a_1,a_2,\ldots,a_n\}$:

```
for j = 1 to n do p_j := j
R = φ
for j = 1 to k do
begin
    r := rand(k,j); (* produces uniformly distributed
          random integer in range k thru' r *)
    R := R + {a_p_r };
    p_r := p_j;
end;
```

Figure 8.2 Generating a random subset

It can be shown (see Reingold, Nievergelt, Deo (1977)) that using such a procedure ensures that each of the $\binom{n}{k}$ k-element subsets has an equal probability of being generated. The procedure is applied sequentially to create the required number of sets.

To perform partitioning on m sets $V_1, V_2, \ldots, V_m$ we do so pairwise, using TWO_WAY_PARTITION $(V_i, V_j)$ on every pair of sets $V_i, V_j$, as long as one of $V_i, V_j$ has changed since the last comparison. A pass is made through all $\binom{n}{2}$ pairs of sets until no more pairwise improvements can be found. Experiments have shown that approximately 95% of the possible improvement occurs during the first two complete iterations, but it is difficult to determine a priori how many iterations will be required by a particular problem.

While not guaranteeing to find the optimal partition, the Kernighan and Lin method performs well, in reasonable computational time. It identifies natural clusters well (if they exist), which is especially advantageous when using DELTAHEDRON; it also consistently produces very similar solutions from different initial configurations. It appears particularly suited to smaller numbers of large sets (rather than vice versa), so would be more applicable to handling layout problems for medium-rise buildings.

STEP (b): Assigning the partition to the building floors

Once our partition of modules to sets has been established, we must now assign these sets to the floors of the building in a manner aimed at minimizing overall transportation cost.

For the case of interfloor travel we assume a fixed-charge model of the form

$$t_{kr} = \alpha + \beta|F(k) - F(r)|$$

where $t_{kr}$ = time to travel from floor F(k) to floor F(r). F(x) is the floor number of facility x, (as mentioned in the formulation of problem MULTI_FLOOR) and $\alpha, \beta$ are constants. This is based upon the existence of only one elevator core.

Justification for this definition is as follows: the components of vertical transit are

(i) waiting time at floor k, a (approximated as a constant for each floor)

(ii) stoppage times at each intermediate floors, b (assumed constant)

(iii) travel times between each pair of intermediate floors, c (assumed constant)

If we assume elevators stop at all floors between k and r, this gives a total transit time of

$$\begin{aligned} t_{kr} &= a + b\,(|k - r| - 1) + c|k - r| \\ &= a - b + (b + c)|k - r| \\ &= \alpha + \beta|k - r| \end{aligned}$$

The values a, b and c will be parameters dependent upon particular elevator performance, and could be modified to approximate actual configurations (for example, relaxing the assumption of stopping at all intervening floors to provide for stopping at, say, every second floor). The case of express elevators would require a unique value of $\beta$ being supplied to travel to and from level 1.

The advantage of using the fixed-charge model is that the values of $\alpha$ and $\beta$ become superfluous in the calculation of the relative costs of any set-floor assignment (if we exclude the availability of express transit), as we now demonstrate.

Consider a partition into m sets $V_1, V_2, \ldots, V_m$. For two sets $V_k$ and $V_r$, with

$$V_k = \{x_1, x_2, \ldots, x_r\}$$
$$V_r = \{y_1, y_2, \ldots, y_s\},$$

where the $x_i$'s and $y_j$'s represent facility modules, and r is not necessarily equal to s, define

$$\text{intra-cost } [V_k, V_r] = \sum_{i=1}^{r} \sum_{j=1}^{s} w_{x_i y_j},$$

representing a measure of the communication between the two sets.

Define an assignment $\phi$ of the m sets to m floors by

$$\phi(V_i) = k \iff \text{set } V_j \text{ is assigned to floor } k$$

Then the value of the assignment $\phi$ is given by

$$\sum_{i=1}^{M} \sum_{j=i+1}^{M} \text{intra-cost}[V_i, V_j](\alpha + \beta|\phi(V_i) - \phi(V_j)|)$$

$$= \alpha \sum_{i=1}^{M} \sum_{j=i+1}^{M} \text{intra-cost}[V_i, V_j]$$

$$+ \beta \sum_{i=1}^{M} \sum_{j=i+1}^{M} \text{intra-cost}[V_i, V_j]|\phi(V_i) - \phi(V_j)|$$

The first term in this expression is constant, so that the relative rankings of assignments are dependent only upon the second term. $\beta$ acts merely as a scaling factor, and so may be ignored in determining the minimal cost assignment. This simplification, of course, does not provide us with the actual cost of the derived assignment; for this to be obtained, the values of the vertical circulation parameters, $\alpha$ and $\beta$ must be known. In terms of our graph theoretic heuristic, these values become important only when the cost of the completed layout is required.

For problems incorporating a medium range number of floors, say 10 or fewer, the minimal cost assignment may be found by enumeration. In order to reduce the computational

effort required, we generate the sequence of required permutations by using the technique of minimal interchange (see Reingold, Nievergelt and Deo (1977)). Each permutation differs from the last by the transposition of only two adjacent elements. Using this technique, the relative cost of each successive permutation may be derived from the most recently found cost via a constant number of additions and subtractions i.e.: the cost change resulting from the swapping of the adjacent elements.

We note two special cases which may occur, but which have not been explicitly considered in our multi-floor implementation. The first is the provision for basement levels, i.e.: floors below the floor containing the entrance facility. Given that facilities appropriate to such levels are usually specialised (service, etc), prelocation would probably create little bias in the solution; the basement floor allocations could then be ignored during the heuristic application, until the layout phase. If fuller flexibility is desired (for example, in the case where topographical requirements imply the necessity of having the entrance on a floor, k say, other than floor 1), permutations will then have to be generated by fixing element k, and computing inter-floor transit costs relative to that position.

A second, and perhaps more common case, is that of differing floor sizes. If we assume the condition

area of floor k $\leq$ area of floor r <=> k > r

then some immediate consequences are

(i) if the first inequality is <u>strict</u>, then no feasible permutations are possible

(ii) if the entrance facility is to be situated on

the ground floor, then, during the pair-wise partitioning procedure, the modules comprising this facility must be assigned to one of the sets of the sets of largest cardinality, and thereafter become non-viable candidates for inter-set swapping. (Such a modification to the partitioning process would be easily implemented). For the provision of the entrance facility on the $k^{th}$ floor, preposition of the corresponding modules would be required in the k-th largest set.

(iii) Suppose the group of partitioned sets is ordered according to decreasing cardinality. Without loss of generality, suppose that this order is $V_1, V_2, \ldots, V_M$, with the exterior facility present in set $V_1$. Group the sets $V_2, V_2, \ldots, V_M$ according to cardinality:

$$V_i \in \text{Group } j \iff |V_i| = j, \quad i=2,\ldots,M$$

Then, at the permutation-generation phase, permutations within each group j need only be considered. The minimal-interchange technique may be invoked as a subroutine, to act upon each group in succession, to produce the desired permutations.

As mentioned above, explicit enumeration is viable only for problems involving at most ten floors. For larger problems we suggest a simple greedy approach, as indicated in Figure 8.3. Beginning with the usual initial allocation to floor 1, sets are sequentially assigned to floors on the

```
Procedure GREEDY_FLOOR_ASSIGNMENT;
begin
    V := {V_1,...,V_M};
    identify set V_r as set containing the "entrance" facility
    ASSIGN(1) := r; (* set V_r assigned to floor 1 *)
    FLOOR := 2;
    ASSIGNED_SET := {V_r};
    repeat
        for each V_i   V-ASSIGNED_SET do
            calculate total interconnection with ASSIGNED_SET,
                taking floor separation into account;
        select V_j with greatest interconnection;
        ASSIGN(FLOOR) := j;
        FLOOR := FLOOR + 1;
        ASSIGNED_SET := ASSIGNED_SET + {V_j}
    until ASSIGNED_SET = V;
    output ASSIGN;
end;
```

Figure 8.3 Assigning the partition greedily

basis of the ranked total interconnection with floors/sets already assigned, weighted according to inter-floor distances. Any solution will therefore be biased with respect to the initial few allocations, but such biasing may be reducible by subsequent pairwise interchanges (which lead to improved overall interconnection costs).

In the case of DELTAHEDRON, the floor-assignment basis of inter-floor travel cost in the above form is no longer appropriate; in DELTAHEDRON we are dealing with adjacency scores. We adopt a scheme akin to the philosophy of N_BOUNDARY_DELTAHEDRON by introducing a simple scaling system

to reflect inter-floor adjacency benefits. If two facilities, i and j, are separated by k (> 0) floors, then we arbitrarily define

$$w_{ij}^{k} = \frac{1}{k} w_{ij} \qquad (\text{or } \frac{1}{k^2} w_{ij})$$

(The actual form of the proportionality function may best be considered as a problem-dependent parameter of the system, but the indicated values are physically appealing.) Hence, the total inter-floor adjacency between any two floors $f_i$ and $f_j$, which are $d(f_i,f_j)$ floors apart, becomes

$$\text{inter-cost}(f_i,f_j) = \sum_{i \in f_i} \sum_{j \in f_j} w_{ij}^{d(f_i,f_j)} .$$

This value represents an 'average' adjacency score irrespective of the final facility positions relative to the vertical circulation zone, so is only an approximation.

For any particular set-floor assignment we may then use these inter-cost scores to evaluate the relative performance of the configuration in either an enumerative or greedy fashion in order to maximize the overall pair-wise adjacency scores.

STEP (d): Constructing the block plan: an outline:

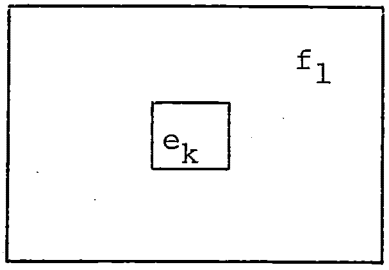
In this section we propose an approach to floor layout in the context of multi-floor building design. The ideas presented here ultimately require modifications to the methods introduced in Chapter 7; implementation of these modifications is considered beyond the scope of the dissertation and has not been attempted.

As an initial stage in constructing a block plan it is useful to coalesce the subfacility modules of each floor into their original facility form. If a facility is split between

two or more floors in the final partition then a 'new' facility is created for each section of it, with each endowed with journey numbers (the $v_{ij}$ values) proportional to its areal fraction. It is possible to solve the problem without coalescing, but the danger would then exist of irregularly-shaped facility generation, and, if the elemental module area is small, processing time usage would be expensive. An excessively large number of modules may also reduce the effectiveness and accuracy of the SUPER_DELTAHEDRON distance approximation.

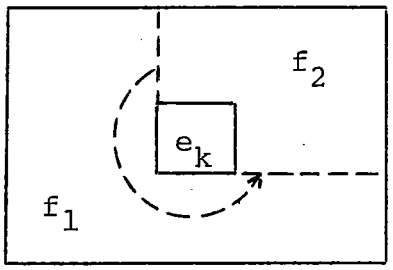
Two further facilities are now considered for each floor, k: the elevator, $e_k$, and the exterior, facility $\varepsilon_k$. As initial input data for SUPER_DELTAHEDRON we assign an artificial number of (scaled) trips from each facility to the exterior that reflects some measure of proximity desirability in terms of physical aspects such as lighting provision and heat-loss minimization (rather than ingress and egress); these chosen levels must be appropriate and consistent within the cost minimization framework. There are two straightforward approaches for introducing facilities $e_k$ and $\varepsilon_k$ - construction of the underlying adjacency graph may begin with either as part of the initial tetrahedron. (We suppose $|V_k| > 3$; otherwise standard constructions will simply be of the form indicated in Figure 8.4). If we choose to initialise the construction with respect to $e_k$, then the other members of the initial tetrahedron will comprise those three vertices with maximal inter-floor communication - these values have been determined previously in the floor-assignment routine calculations. The remaining vertex insertion

(a)



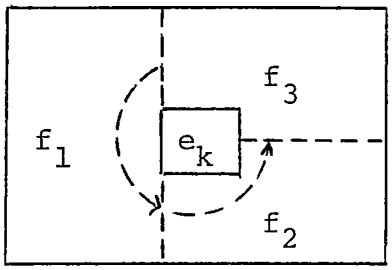

$|V_k| = 1$

(b)




$|V_k| = 2$

(c)




$|V_k| = 3$

Figure 8.4 Example layouts for the cases $|V_k| \leq 3$. Division is concomitant with area.

order would also follow these rankings. For simplicity, we also assume that the elevator is centrally located, with given area (which may also include the corresponding service facilities which frequently are part of standard elevator cores). This assumption could be modified to, say, 'n unit modules from the exterior', with consistent orientation.

Problems arise, however, in starting with $e_k$: physically, perhaps only four facilities may be adjacent to it, and this may inhibit flexibility in the construction - see Figure 8.5. It is also difficult to determine where to place $\varepsilon_k$ in the insertion order. For these reasons we prefer to create an initial tetrahedron including $\varepsilon_k$. This also means that we could more easily take advantage of the layout method of Chapter 7.

The revised methodology means that the vertex insertion order will revert to the (restricted) greedy-cum-column-sum approach adopted in the last chapter. Facility $e_k$ is introduced after the final facility-vertex insertion, into the triangle whose inter-floor communication cost is maximal; this implies that $\deg(e_k) = 3$, a figure which may be increased to 4 if suitable improvement methods show it to be warranted. Implicit in this approach is the assumption that the set partitioning phase has identified most natural clusters in a problem instance, so that the layout phase for each floor is essentially independent of the other floors in the building with the exception of at most three or four facilities which are best placed adjacent to the elevator. Likely candidates in the final insertion triangle are sections of split facilities; for all other facilities on a floor, the dominant inter-floor cost is usually the vertical movement component,

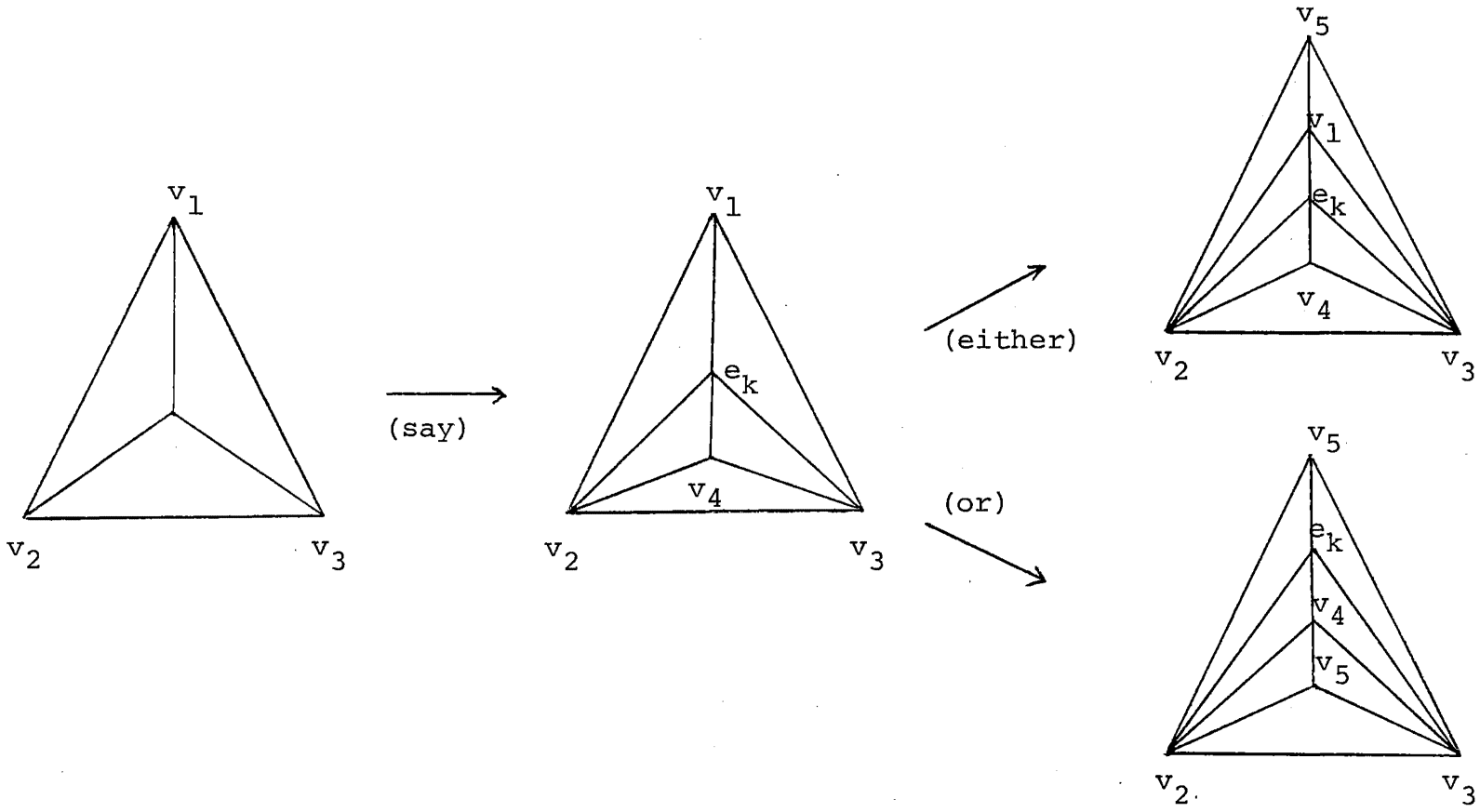



Figure 8.5 Insertion of vertices $v_4, v_5$ into tetrahedron $\langle e_k, v_1, v_2, v_3 \rangle$

so that the within-floor adjacency specification is more important. Departmental autonomy provides further justification for the simplification.

Not any triangle in the adjacency graph may be chosen for the insertion of $e_k$ - for example, proximity to facility $\varepsilon_k$ is forbidden in this model (exterior elevator configurations are not unknown, however, but tend to be inefficient under the criterion of global minimization (Tabor (1970))). In the terminology of Chapter 7 we allow only insertions of Type I and Type III, so that $e_k$ will always be part of an inner distance class. It may be possible to also allow insertion in a triangle consisting of vertices of $DC_2$ only - hence creating a single element class, $DC_3$. Either case will correspond to the general layout form of Figure 8.6, but the configuration will be attained via different routes. If $e_k \in DC_2(k)$ then it will be necessary to modify the SPLAN routine to handle its prelocation; if $e_k \in DC_3$, then $DC_2(k)$ must be laid out in a form akin to the $DC_1(k)$ ring. As mentioned previously, the implementation details of these modifications have not been considered.

For the DELTAHEDRON heuristic, the method will parallel that outlined above. The relationship ratings of facilities relative to the exterior need not be artificially defined (since they will exist as input); the objective is now, of course, maximization. Given that facility areas are not considered in the DELTAHEDRON approach, the required modifications to the PLAN routine will undoubtedly be simpler that those to SPLAN. The design produced, however, is unlikely to be as practically oriented.

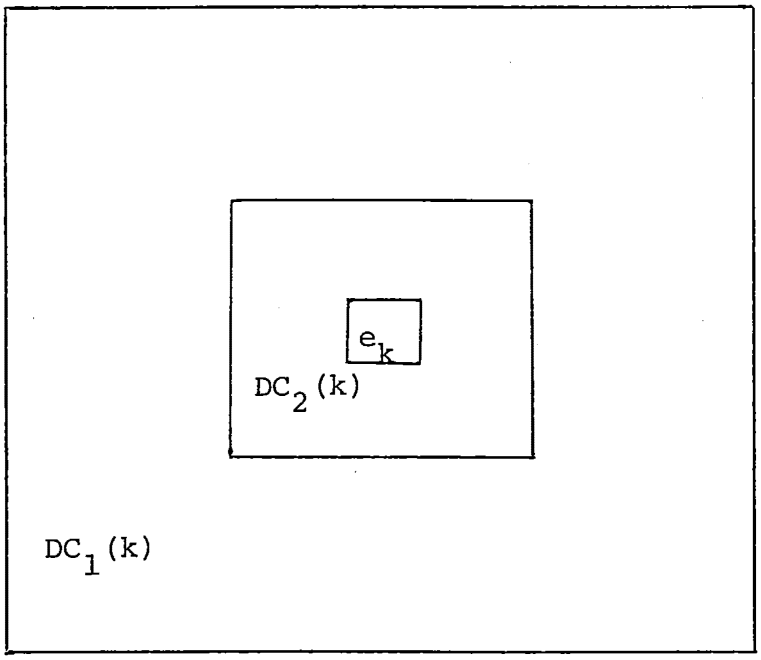



Figure 8.6 A stylized floor layout scheme

## 8.3 Computational experience: partitioning and assignment

Few practical problems exist in the literature with which we may compare our initial phases of multi-floor layout. Carter and Whitehead (1975) present a 30 facility, 3-floor problem based on the design of a Police Station; cost data is in the form of an "association matrix" of (unscaled) journeys. No facilities are prelocated. Unit module areas are provided. The data is summarized in Figure 8.7.

Figure 8.8 displays the facility partitions and floor assignments obtained by

(i) "Genopt" - Portlock and Whitehead (1974)

(ii) "Clust" - Carter and Whitehead (1975)

(iii) the method of section 8.2.

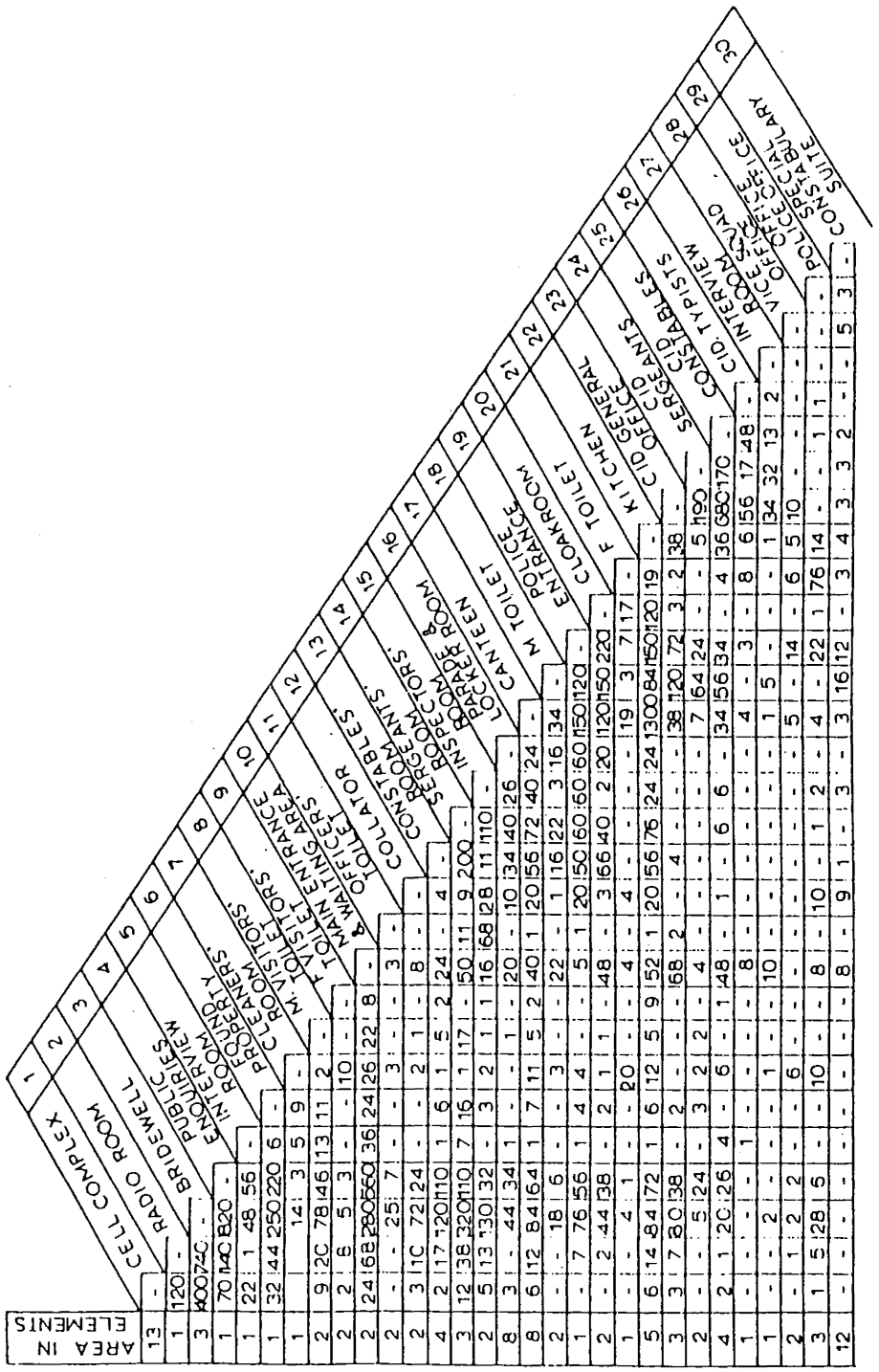



Figure 8.7 Relationship chart and areas for the Police Station example

(i)

| FLOOR 1 | FLOOR 2 | FLOOR 3 |
|---|---|---|
| 1 | 11 | 5 |
| 2 | 12 | 7 |
| 3 | 15 | 8 |
| 4 | 16 | 9 |
| 6 | 18 | 23 |
| 10 | 19 | 27 |
| 13 | 20 | 28 |
| 14 | 21 | 30 |
| 17 | 22 | |
| | 24 | |
| | 25 | |
| | 26 | |
| | 29 | |

(ii)

| FLOOR 1 | FLOOR 2 | FLOOR 3 |
|---|---|---|
| 7 | 1 | 23 |
| 9 | 2 | 24 |
| 12 | 3 | 25 |
| 16 | 4 | 26 |
| 17 | 5 | 27 |
| 18 | 6 | 28 |
| 19 | 8 | 30 |
| 20 | 10 | |
| 21 | 11 | |
| 22 | 13 | |
| 29 | 14 | |
| | 15 | |

(iii)

| FLOOR 1 | FLOOR 2 | FLOOR 3 |
|---|---|---|
| 5 | 1 | 13 |
| 6 | | |
| 7 | 2 | 14 |
| 8 | | |
| 10 | 3 | 16 |
| 11 | | |
| 12 | 4 | 17 |
| 15 | 9 | 18 |
| 21 | | |
| 26 | 18 | 19 |
| 27 | 23 | 20 |
| 29 | 24 | 22 |
| 30 | 25 | |
| | 28 | |

Figure 8.8 Partitions for the Police Station problem

The corresponding solution values are

(i) 56182

(ii) 47249

(iii) 51377 (49377)

It should be noted that solutions (i) and (ii) are produced by assuming floor areas of 35 modules each, allowing some "slack", whereas solution (iii) is for the 'tight' 32 module areas. When this restriction is relaxed, facility 18 coalesces onto floor 3, with a corresponding cost of 49377 (inter-module costs were set at 1000). The "Clust" method produces the best solution, but the present approach performs well. Solution time is not available for (i); for (ii) the quoted figure is ~20 secs (ICL 1906S); at 14.57 seconds (Burroughs B6930), the graph theoretic model compares favourably.

The present implementation is unfortunately limited to problems of fewer than 127 modules in total - this is due to "page-faulting" under the Burroughs Pascal compiler. This means that the CRAFT_3D example of Cinar (1975) and that of Johnson (1982) cannot be encoded and compared.

Several random problems were generated to evaluate the performance of the partitioning and assignment phases. Tables 8.1 and 8.2 give a representative sample of the results for problems of various sizes. As expected, examples with 10 floors represent the limit for the enumeration scheme floor-assignment; this phase is the dominant factor in the processing time of all cases studied. Problems with large numbers of modules (typically the result of higher areal variance) require considerably more computational effort because more

pairwise comparisons must be evaluated during partitioning. Over all the problems generated, the cost of the optimal floor-assignment average an 8.1% improvement over that of a random assignment, justifying the extra effort. Because of the prohibitive cost of computing, the performance of the greedy approach (applicable to larger problems) has not yet been tested.

| # Facilities | # Modules | # Floors | CPU-time (secs) |
|---|---|---|---|
| 5 | 50 | 5 | 6.45 |
| 20 | 100 | 10 | 25.51 |
| 30 | 96 | 3 | 14.57 |
| 40 | 40 | 7 ** | 4.14 |
| 50 | 50 | 10 | 8.43 |
| 100 | 100 | 10 | 28.40 |

Table 8.1 Partitioning

** This example had tapering floor sizes (7,7,6,6,5,5,4)

| # Facilities | # Modules | # Floors | CPU-time (secs) |
|---|---|---|---|
| 8 | 20 | 4 | 1.56 |
| 16 | 44 | 8 ** | 13.64 |
| 16 | 44 | 8 ** | 14.89 |
| 16 | 44 | 8 ** | 15.46 |
| 20 | 72 | 9 *** | 93.71 |
| 30 | 96 | 6 | 20.81 |
| 30 | 96 | 8 | 30.08 |
| 25 | 50 | 10 | 852.26 |
| 25 | 100 | 10 | 882.04 |

Table 8.2 Partitioning and Floor-Assignment*

* using the enumeration routine

** unequal floors (6,6,6,6,5,5,5,5)

*** unequal floors (10,10,9,9,8,7,7,6,6)

# CHAPTER 9

## SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

In this dissertation we have considered several graph theoretic constructional techniques for various problems in Facilities Layout. Since the basic formulation, whether in the context of Quadratic Assignment or Graph Theory, is $\mathcal{NP}$-complete, we cannot hope to discover efficient algorithms for practical sized problems (unless $\mathcal{P}=\mathcal{NP}$), so we turn to heuristics.

For the case of maximizing the relationship chart scores of pairs of adjacent facilities, we compared three heuristics in Chapter 3. IMPROVED_DELTAHEDRON proved to be the most effective method in terms of both solution quality and efficiency if resources are limited, although the solutions obtained by GREEDY were often superior. Sample bounds on performance were also given. The procedure WHEEL_GENERATION was introduced as a promising alternative; both it and WHEEL_EXPANSION have the advantage of being able to form the basis for the enumeration of elementary architectural configurations.

The methods of Chapter 3 may produce an adjacency graph whose structure is impractical because it contains vertices of high degree. A simple way of alleviating this problem was discussed in the following Chapter, where we considered the more realistic objective of the maximization of relationship chart scores for <u>all</u> pairs of facilities.

Heuristic solutions to this problem were achieved via modification of DELTAHEDRON and GREEDY, the greater complexity of the formulation requiring the introduction of an efficient updating form of a slight simplification of GREEDY. Unfortunately the type of data necessary as input to the methods discussed would be difficult to obtain, possibly reducing the value of this extension.

Efficient updating was pursued in Chapter 5. Here we addressed the augmentation problem: given a planar graph $G = (V,E)$ with $|E| < 3|V|-6$ and an edge $e \notin E$, is $G + e$ planar? This was motivated by the need to improve the performance of GREEDY without the need for restriction of the rationale. Although the basis of the approach was the procedure of Fisher and Wing rather than that of Hopcroft and Tarjan, it is felt that the cluster-based methodology developed could provide a competitive alternative implementation.

Previously mentioned graph theoretic techniques did not incorporate facility areas explicitly. Chapter 6 discussed a possible framework for doing so, in terms of an approximation to rectilinear travel embedded within the DELTAHEDRON heuristic. The objective becomes the minimization of total transportation cost. The maximum degree of any vertex in the constructed graph was constrained to an area-dependent level, to ensure that the corresponding layout will be more practical and more easily achievable. Computational experience suggests that this so-called SUPER_DELTAHEDRON approach is the most promising of the methods described.

In Chapter 7 we developed methodology for the systematic construction of a block plan corresponding to a restricted class of adjacency graphs generated via two versions of constrained DELTAHEDRON (modifications of both DELTAHEDRON and SUPER_DELTAHEDRON). The procedure was designed to handle problems involving up to 25 facilities and to be implementable on a 64k microcomputer. A very brief performance comparison with CRAFT and CORELAP displays deficiencies which exist in the approach; for the example problem these were rectifiable by simple manual manipulation. Such ornamentation is common to all existing methods, and in the present case is partly due to the specialisation required to permit microcomputer compatability. It also serves to highlight the preliminary nature of the block plan construction investigation. Even in its initial form, however, the potential power of the method appears very promising.

The final chapter provided a basis for extending the range of applicability of the SUPER_DELTAHEDRON heuristic to multi-floor layout. Kernighan and Lin's partitioning method is used to obtain a partition of the vertices into sets with minimal inter-set communication; the set sizes are determined by the building floor areas, and these sets are assigned to the floors in a cost-minimizing configuration based on a fixed-charge model. Each floor may then be laid out essentially independently of the others, with the vertex insertion order slightly modified to take account of proximity to the exterior facility and the level of inter-floor communication of a few crucial facilities. It is critical that the elevator facility be consistently placed

on each floor; until the block plan procedure of Chapter 7 is suitably refined to handle this phase, and more general layouts, implementation of the layout phases will not be attempted.

The preceding summary indicates only some of the uses of Graph Theory in facilities layout. Simplicity and versatility in modelling are very desirable features; this dissertation has attempted to explore and employ these facets of the two-phase graph theoretic approach. We have demonstrated that the methods can be efficient and effective for the problem of determining highly weighted adjacency graphs, and that a simple relative performance measure may be easily obtained. Problems larger than human subjects would cope with may be dealt with easily. Performance guarantees are also available for some cases. Of course, as is the case with manual methods, the results are only as good as the data - techniques of estimation of relationship chart scores and transportation costs still require refinement.

An important development has been the first step in the systematic generation of a block plan from the adjacency graph. The so-called "polyomino assembly procedure" has long been recognised as a difficult problem; our method provides a solution for a representative subset of the general case, and may be accommodated within minimal computing capability. Multi-floor layout is a natural extension, and potentially allows the solution of problems larger than are currently possible.

While a purely combinatorial viewpoint could have been taken throughout the thesis, we have chosen a more

practical stance. Implementation should always be the first phase of the Operations Research approach, and we have tried to adhere to this in the design of the methods and the formulation of the problems. Whether we have been successful will be seen in the further use of the techniques in a real-world application. (See Foulds and Tran (1984) for an example library layout.) We feel that the results of this ultimate test will justify our faith in graph theoretic techniques for facilities layout.

Clearly there is much still to be done in this topic area. We conclude with several comments and suggestions for future research on problems that were considered outside the scope of the thesis.

Further analysis is required to determine the performance characteristics of the heuristics. A worst case analysis of SUPER_DELTAHEDRON would be useful, but perhaps an average-case analysis of DELTAHEDRON would be more instructive, to give some insight into why its performance is usually very good, when it has shown that the method may perform arbitrarily badly.

Sensitivity analysis could be used to provide a measure of the robustness of a solution with respect to a given heuristic. Modifications to entries in the relationship matrix to create a scenario sequence of adjacency graphs would also allow evaluation of the appropriateness of the weightings used, and help to eliminate previously undetected and unwanted bias in the solution. These suggestions offer a further alternative to the ornamentation steps available through the improvement routines.

A further heuristic for problem ADJACENCY may be derivable from the work of Christofides, Galliani and Stefanini on the Lagrangian formulation. A greedy technique based on edges and triangles, with periodic heuristic updating of the Lagrange multipliers using subgradient techniques has already been outlined. The performance of this and WHEEL_GENERATION should make an interesting comparison with GREEDY and DELTAHEDRON.

In our discussion of efficient updating procedures we considered the Fisher and Wing representation. Given that the Hopcroft and Tarjan procedure is somewhat more efficient, it is possible that wholesale changes forced upon the depth-first-search representation by the addition of an edge at the augmentation step may occur only infrequently for any given problem instance, so that the average-case expected complexity may not be too adversely affected. A characterisation of the types of incremental changes to (and their effects on) the DFS-tree would be a first step in algorithm development. Thus, despite the rather negative comments of Chapter 5, an efficient updating form (not based upon the cluster representation) may then allow considerable improvement to GREEDY. Similarly, when full details of the "Left-Right" algorithm of de Fraysseix and Rosenstiehl are known, similar adjustments could prove possible. The observations of Yeh (1982) on improving general planarity testing may also be employed to advantage.

As was mentioned in Chapter 7, further analysis is required of the block plan generation mechanism, and its relationship to the SUPER_DELTAHEDRON metric. Extensive empirical computational comparison with CRAFT, its

derivatives and competitors, is required to assess its performance and characterise the types of problems to which it is best suited. If possible, a procedure which could handle any maximal planar graph would be desirable, but the problems encountered in allowing the limited scope already described indicate the need for a myriad of special cases or a complete reassessment of the current methodology. (Including orientation desirability for each facility with respect to the cardinal directions would help simplify the problem by reducing the number of feasible configurations.) It also appears that perturbations from the rigid maximal planar graph structure are worthwhile in cases of severe distortion from facility shapes approximating squares. Automation, presumably with interactive enhancement, of these (presently) manual techniques is a logical addition. Provision of communication paths (corridors) in a layout could also be generated using a graph theoretic model based on the rectangular network representing the layout as a post-construction phase - determining such information from the adjacency graph, which at best estimates the spatial location of the facilities, would be difficult and less accurate.

Completion and refinement of the multi-floor layout model will be dependent upon the satisfactory realisation of the successful layout phase mentioned above. Strategies for the constant placement of the vertical circulation zone, and methods of handling more than one such system need to be investigated. Such techniques must be compatible with the single-floor layout procedure.

Finally, we suggest the application of variants of

the techniques described to one of the newer aspects of facilities design - flexible manufacturing.

# REFERENCES

AHO, A.V., HOPCROFT, J.E. and ULLMAN, J.D. (1974) "The Design and Analysis of Computer Algorithms", (Addison-Wesley, Reading, Mass.).

ARCHER, B. (1963) "Planning accommodation for hospitals and the transportation problem technique" (Editorial), Archit. J. 138, 139-142.

BAYBARS, I. (1982) "Generation of Floor Plans with circulation spaces", Env. and Planning B. 9, 445-456.

BAYBARS, I. and EASTMAN, C.M. (1980) "Enumerating architectural arrangements by generating their underlying graphs", Env. and Planning B. 7, 289-310.

BAZARAA, M.S. and ELSHAFEI, A.N. (1979) "An exact branch-and-bound procedure for the Quadratic-Assignment Problem", N.R.L.Q. 26, 109-121.

BLOCH, C.J. and KRISHNAMURTI, R. (1978) "The counting of rectangular dissections", Env. and Planning B. 5, 207-214.

BLOCK, T.E. (1978) "FATE - a new construction algorithm for facilities layout", J. Eng. Prod. 2, 111-120.

BLOCK, T.E. (1979) "On the complexity of facilities layout problems", Mgt. Sci. 25, 280-285.

BOND, A. (1971) "CIRCE: a layout generating program", University of Bristol, Dept. of Architecture W.P. 2/71.

BONDY, J.A. and MURTY, U.S.R. (1976) "Graph Theory with Applications", (Macmillan, London).

BOOTH, K.S. and LUEKER, G.S. (1976) "Testing for the consecutive ones property, interval graphs, and graph planarity, using PQ-tree algorithms", J. Comp. Syst. Sci. 13, 335-379.

BOWEN, R. and FISK, S. (1967) "Generation of triangulations of the sphere", Math. Comp. 21, 250-252.

BOX, G.E.P. and MULLER, M.E. (1958) "A note on the generation of random Normal deviates", Ann. Math. Stats. 29, 610-611.

BROOKS, R.L., SMITH, C.A.B., STONE, A.H. and TUTTE, W.T. (1940) "The dissection of rectangles into squares", Duke Math J. 7, 312-340.

BUFFA, E.S. (1976) "On a paper by Scriabin and Vergin", Mgt. Sci. 23, 104.

BUFFA, E.S., ARMOUR, G.C. and VOLLMAN, T.E. (1964) "Allocating facilities with CRAFT", Harv. Bus. Rev. 42, 136-159.

BURKARD, R.E. and STRATMANN, K.-H. (1978) "Numerical investigations on quadratic assignment problems", NRLQ 25, 129-148.

CARRIE, A.S., MOORE, J.M., ROCZNIAK, M. and SEPPANEN, J.J. (1978) "Graph Theory and Symbolic Processors for Computer Aided Facilities Design", Omega 6, 353-361.

CARTER, D.J. and WHITEHEAD, B. (1975) "The use of cluster analysis in Multistorey layout planning", Building Sci. 10, 287-296.

CHESTON, G.A. (1976) "Incremental algorithms in graph theory", Dept. of Comp. Sci. Tech. Rep. 91, University of Toronto.

CHESTON, G.A. and CORNEIL, D.G. (1982) "Graph property update algorithms and their application to distance matrices", INFOR 20, 178-201.

CHRISTOFIDES, N., GALLIANI, G. and STEFANINI, L. (1980) "An algorithm for the Maximal Planar Graph problem based on Lagrangean Relaxation" (Imperial College, London).

CHYUTIN, M. and MITTWOCH, D.Z. (1979) "Optimal allocation of facilities in high-rise buildings", Build. and Env. 14, 235-239.

CINAR, U. (1975) "Facilities Planning: a systems analysis and space allocation approach", in Eastman, C.M. (ed) "Spatial synthesis in computer-aided building design" (Applied Science, London), 19-40.

COLEMAN, D.R. (1977) "Plant Layout: computers versus Humans", Mgt. Sci. 24, 107-112.

COOK, S.A. (1971) "The complexity of theorem-proving procedures", Proc. 3rd Ann. ACM Symp. on Theory of Computing, ACM New York, 151-158.

CROSS, N. (1977) "The Automated Architect" (Pion Ltd, London).

DAELLENBACH, H.G., GEORGE, J.A. and McNICKLE, D.C. (1983) "An introduction to Operations Research Techniques (2nd Ed)", (Allyn and Bacon, Boston).

DANTZIG, G.B. (1967) "All shortest routes in a graph", Theory of Graphs, Proc. Int. Symp., Rome, July 1966 (Gordon and Breach, N.Y.).

DÉMOUCRON, G., MALGRANGE, Y. and PERTUISET, R. (1964) "Graphes planaires et construction des representations planaires topologiques", Rev. Franc. Recher. Oper. 8, 33-47.

DE FRAYSSEIX, H. and ROSENSTIEHL, P. "A depth-first-search characterisation of planarity", Proc. 1981 Cambridge Combinatorial Conf. (B. Bollobas, ed.).

DYER, M.E., FOULDS, L.R. and FRIEZE, A.M. (1983) "The analysis of some graph theoretic heuristics for the facilities layout problem" (Dept. of Computer Science and Statistics, Queen Mary College, London).

EADES, P., FOULDS, L.R. and GIFFIN, J.W. (1982) "An efficient heuristic for identifying a maximum weight planar subgraph", Combinatorial Mathematics IX, Lecture Notes in Math. No. 952, (Springer-Verlag, Berlin), 239-251.

EARL, C.F. (1981) "Enumerating architectural arrangements: comment on a recent paper by Baybars and Eastman", Env. and Planning B. 8, 115-118.

EBERHARD, V. (1891) "Zur Morphologie der Polyeder" (Teubner, Leipzig).

EULER, L. (1752) "Demonstratio nonnullarum insignium proprietatum quibus solida hedris planis inclusa sunt praedita," Novi Comm. Acad. Sci. Imp. Petropol 4, 140-160 (Opera Omnia 26, 94-108).

EVEN, S. (1979) "Graph Algorithms", (Pitman, London).

FISHER, G.J. and WING, O. (1966) "Computer recognition and extraction of planar graphs from the incidence matrix", IEEE Trans. on Circ. Th. CT-13, 154-163.

FLEMMING, U. (1978) "Wall representation of rectangular dissections and their use in automated space allocation", Env. and Planning B. 5, 215-232.

FISHER, M.L., NEMHAUSER, G.L. and WOLSEY, L.A. (1979) "An analysis of approximations for finding a maximum weight hamiltonian circuit", O.R. 27, 799-809.

FLOYD, R.W. (1962) "Algorithm 97: Shortest Path", Comm. A.C.M. 5, 345.

FOULDS, L.R. (1983) "Techniques for facilities layout: deciding which pairs of activities should be adjacent", Mgt Sci. 29, 1414-1426.

FOULDS, L.R., GIBBONS, P.B. and GIFFIN, J.W. (1984) "Facilities layout adjacency determination: an experimental comparison of graph theoretic heuristics", Oper. Res. (to appear).

FOULDS, L.R. and GIFFIN, J.W. (1983) "A Graph Theoretic method for minimizing total transportation cost in Facilities Layout", A.I.I.E. Transactions (submitted).

FOULDS, L.R. and ROBINSON, D.F. (1976) "A strategy for solving the Plant Layout Problem", Op. Res. Quart. 27, 845-855.

FOULDS, L.R. and ROBINSON, D.F. (1978) "Graph theoretic heuristics for the plant layout problem", Int. J. Prod. Res. 16, 27-37.

FOULDS, L.R. and ROBINSON, D.F. (1979) "Construction properties of combinatorial deltahedra", Discr. Appl. Math. 1, 75-87.

FOULDS, L.R. and TRAN, H.V. (1984) "Library layout with Graph Theory", College and Research Libraries (to appear).

FRANCIS, R.L. and WHITE, J.A. (1974) "Facility layout and location: an analytical approach", (Prentice-Hall, N.J.).

GAREY, M.R. and JOHNSON, D.S. (1979) "Computers and Intractability: A guide to the theory of NP-Completeness", (Freeman, San Francisco).

GAVETT, J.W. and PLYTER, N.V. (1966) "The optimal assignment of facilities to locations by branch and bound", Oper. Res. 14, 210-232.

GAWAD, M.T.A. and WHITEHEAD, B. (1976) "Addition of communication paths to diagrammatic layouts", Build. and Env. 11, 249-258.

GERO, J.S. (1976) "Note on "Synthesis and optimization of small rectangular floor plans" of Mitchell, Steadman and Liggett", Env. and Planning B. 4, 81-88.

GIFFIN, J.W. and FOULDS, L.R. (1983) "Improved Graph Theoretic techniques for facilities layout", Dept. of Economics, University of Canterbury.

GIFFIN, J.W. and FOULDS, L.R. (1984) "Efficient Graph Planarity Updating Tests", Proc. 11th Australasian Conference on Combinatorial Mathematics and Computing, (Utilitas Math., Winnipeg).

GILLEARD, J. (1978) "LAYOUT - a hierarchical computer model for the production of architectural floor plans", Env. and Planning B. 5, 233-241.

GRASON, J. (1970) "A dual linear graph representation for space-filling location problems of the floor plan type", in "Emerging methods in environmental design and planning" (G.T. Moore, ed.), 170-177.

GRÜNBAUM, B. (1967) "Convex Polytopes" (J. Wiley and Sons, N.Y.).

HARARY, F. (1969) "Graph Theory", (Addison-Wesley, Reading, Mass.).

HASHIMSHONY, R., SHAVIV, E. and WACHMAN, A. (1980) "Transforming an adjacency matrix into a planar graph", Build. and Env. 15, 205-217.

HILLIER, F.S. (1963) "Quantitative tools for plant layout", J. Ind. Eng. 14, 33-40.

HILLIER, F.S. and CONNORS, M.M. (1966) "Quadratic Assignment Problem algorithms and the location of indivisible facilities", Mgt. Sci. 13, 42-57.

HOPCROFT, J.E. and TARJAN, R.E. (1974) "Efficient planarity testing", J.A.C.M. 21, 544-568.

HYAFIL, L. and RIVEST, R.L. (1973) "Graph partitioning and constructing optimal decision trees are polynomial complete problems", Report No. 33, IRIA-Laboria, Rocquencourt, France.

JOHNSON, R.V. (1982) "SPACECRAFT for multifloor layout planning", Mgt. Sci. 28, 407-417.

KARP, R.M. (1972) "Reducibility among combinatorial problems", in R.E. Miller and J.W. Thatches (eds), "Complexity of Computer Computations", (Plenum Press, N.Y.), 85-103.

KERNIGHAN, B.W. and LIN, S. (1970) "An efficient heuristic procedure for partitioning graphs", Bell Syst. Tech. J. 49, 241-307.

KOOPMANS, T.C. and BECKMANN, M. (1957) "Assignment problems and the location of economic facilities", Econometrica 25, 52-76.

KORF, R.E. (1977) "A shape independent theory of space allocation", Env. and Planning B. 4, 37-50.

KORTE, B. (1979) "Approximative algorithms for discrete optimization problems", Ann. Discr. Math. 4, 85-120.

KORTE, B. and HAUSMANN, D. (1978) "An analysis of the greedy heuristic for Independence Systems", Ann. Discr. Math. 2, 65-74.

KRARUP, J. and PRUZAN, P.M. (1978) "Computer-Aided Layout Design", Math. Prog. Study 9, 75-94.

KREJCIRIK, M. (1969) "Computer aided plant layout", Computer Aided Design 2, 7-19.

KURATOWSKI, K. (1930) "Sur le probleme des courbes gauches en topologie", Fund. Math. 15, 271-283.

LAWLER, E.L. (1975) "The Quadratic assignment problem: a brief review", in Roy, B. (ed), "Combinatorial programming: Methods and applications" (D. Reidel, Dordrecht), 351-360.

LEE, R. and MOORE, J.M. (1967) "CORELAP - Computerized Relationship Layout Planning", J. Ind. Eng. 18, 195-200.

LEMPEL, A., EVEN S. and CEDERBAUM, I. (1966), "An algorithm for planarity testing of graphs", Theory of Graphs, Int. Symp. Rome, July 1966 (Gordon and Breach, New York, 1980) 215-232.

LENSTRA, J.K. (1976) "Sequencing by enumerative methods" (Mathematisch Centrum, Amsterdam).

LEVIN, P.H. (1964) "Use of graphs to decide optimum layout of buildings", Archit. J. 140, 809-815.

LEWIS, W.P. and BLOCK, T.E. (1980) "On the application of computer aids to plant layout", Int. J. Prod. Res. 18, 11-20.

LIN, P.C. and GELDMACHER, R.F. (1976) "On the deletion of non-planar edges of a graph", Dept. of Electrical Engineering, Stevens Institute of Technology, Hoboken, N.J.

MARCH, L. and EARL, G.F. (1977) "On counting architectural plans", Env. and Planning B. 4, 57-80.

MITCHELL, W.J., STEADMAN, J.P. and LIGGETT, R.S. (1976) "Synthesis and optimization of small rectangular floor plans", Env. and Planning B. 3, 37-70.

MOORE, J.M. (1976) "Facilities Design with Graph Theory and Strings", Omega 4, 193-203.

MOORE, J.M. and CARRIE, A.S. (1976) "Impact of list processors and graph theory on use of computers for solving facilities design problems", Proc. 3rd Int. Conf. Prod. Res., Amherst (Taylor and Francis, London).

MOSELEY, D.L. (1963) "A rational design theory for planning buildings based on the analysis and solution of circulation problems", Arch. J. Information Library 11, 525-537.

PAPADIMITRIOU, C.H. and STEIGLITZ, K. (1982) "Combinatorial Optimization: Algorithms and Complexity" (Prentice-Hall, N.J.).

PORTLOCK, P.C. and WHITEHEAD, B. (1974) "Three dimensional layout planning", Building Sci. 9, 45-53.

REINGOLD, E.M., NIEVERGELT, J. and DEO, N. (1977) "Combinatorial Algorithms: Theory and Practice", (Prentice-Hall, N.J.).

ROTH, J., HASHIMSHONY, R. and WACHMAN, A. (1982) "Turning a graph into a rectangular floor plan", Build. and Env. 17, 163-173.

ROY, B. (1959) "Transitivité et connexité", C.R. Acad. Sci. Paris 249, 216-218.

SAATY, T.L. and KAINEN, P.C. (1977) "The Four-colour problem: assaults and conquest", (McGraw-Hill, N.Y.).

SAHNI, S. and GONZALEZ, T. (1976) "P-complete Approximation problems", J.A.C.M. 23, 555-565.

SCRIABIN, M. and VERGIN, R.C. (1975) "Comparison of computer algorithms and visual-based methods for plant layout", Mgt. Sci. 22, 172-181.

SCRIABIN, M. and VERGIN, R.C. (1981) "Line dominance, flow dominance and the complexity of facility layout problems", Working paper No.137, Dept. of Management Sciences, Univ. of Waterloo, Ontario, Canada.

SEEHOF, J.M., EVANS, W.O., FREDRICKS, I.W. and QUIGLEY, I.J. (1966) "Automated facilities layout problems", Proc. A.C.M. National Meeting.

SEEHOF, J.M. and EVANS, W.O. (1967) "Automated layout design program", J. Ind. Eng. 18, 690-695.

SEPPANEN, J.J. and MOORE, J.M. (1970) "Facilities Planning with Graph Theory", Mgt. Sci. 17B, 242-253.

SEPPANEN, J.J. and MOORE, J.M. (1975) "String Processing Algorithms for Plant Layout Problems", I.J.P.R. 13, 239-254.

SHAVIV, E., HASHIMSHONY, R. and WACHMAN, A. (1977) "Decomposition of a multi-cell complex - a problem in physical design", Design Meth. Th. (Journal of the D.M.G.) 11, 113-120.

SHAVIV, E., HASHIMSHONY, R. and WACHMAN, A. (1978) "A decomposition-recomposition model for multi-cell systems", Build. and Env. 13, 109-123.

SKUPIEN, Z. (1966) "Locally Hamiltonian and planar graphs", Fund. Math. 58, 193-200.

STEADMAN, J.P. (1976) "Graph theoretic representation of architectural arrangement", in March, L. (ed), "The Architecture of Form" (C.U.P.).

STEINITZ, E. and RADEMACHER, H. (1934) "Vorlesungen uber die Theorie der Polyheder" (Springer, Berlin).

TABOR, P. (1970) "Traffic in buildings: analysis of communication patterns", Centre for land use and built form studies, W.P. No. 19, University of Cambridge.

TEAGUE, L.C. (1970) "Network models of configurations of rectangular parallelopipeds", in Moore, G.T. (ed), "Emerging methods in Environmental Design and Planning", 162-169.

THOMASSEN, C. (1980) "Planarity and duality of finite and infinite graphs", J. Comb. Th. B. 29, 244-271.

TOMPKINS, J.A. and MOORE, J.M. (1978) "Computer Aided Layout: a users guide" (Facilities Planning and Design Monograph Series, No. 1, AIIE, Norcross, Ga).

TRYBUS, T.W. and HOPKINS, L.D. (1980) "Humans vs. Computer algorithms for the plant layout problem", Mgt. Sci. 26, 570-574.

TUTTE, W.T. (1961) "A theory of 3-connected graphs", Indag. Math. 29, 441-455.

VOLLMANN, T.E and BUFFA, E.S. (1966) "The facilities layout problem in perspective", Mgt. Sci. 12, B450-468.

WARSHALL, S. (1962) "A theorem on Boolean matrices", J.A.C.M. 9, 11-12.

WHITE, D.S. (1972) "Mathematical evaluation and optimization of the 3-dimensional hospital layout problem", Proc. E.D.R.A. 3, (23-7-1)-(23-7-7).

WHITNEY, H. (1931) "A theorem on Graphs", Ann. Math. 32, 378-390.

WHITNEY, H. (1932) "Planar Graphs", Fund. Math. 21, 73-84.

WILLOUGHBY, T. (1970) "Computer-aided building plans - a generative approach", Official Arch. Plan. 33, 778-785.

WILLOUGHBY, T. (1971) "Computer use - a direction for computer-aided planning methods", Building 6, 56-60.

WILSON, R. (1972) "Introduction to Graph Theory" (Longman, London).

YEH, D. (1982) "Improved planarity algorithms", BIT 22, 2-16.

# APPENDIX 1

1.1 Program DELTAH.

Written in Microsoft Basic Version 5.0.

Determines the maximal planar adjacency graph via constrained DELTAHEDRON - facility areas are not taken into account.

Sample data included for example of Chapter 7.

1.2 Program PLAN.

Written in Microsoft Basic Version 5.0.

Constructs a block plan corresponding to the output of program DELTAH; it is invoked via a call inside DELTAH, in order to reduce core storage requirements.

```
10 REM DELTAHEDRON HEURISTIC
20 DEFINT I-N
30 DIM BEN%(30,30),ORDER%(30),TRIANG%(56,3),SOLUTION%(30,30)
40 DIM EDGES%(500),R%(20),CARD%(10),DIST%(30),SAVED%(10),HMAXX%(3)
50 DIM CLASS%(10,3),LAYER(10,30),H%(30),PAIR%(3,2),BETW%(3,10)
60 DIM VALID%(30),FUNDTRI%(10),ORIGIN%(10),FUND%(10,2),BADJ%(2)
70 DIM ANO%(30),BEFORE%(30),AFTER%(30,10),KOUNT(10),NUM(30),STORE%(3,3)
80 INPUT "NUMBER OF FACILITIES";N
90 LPRINT "NUMBER OF FACILITIES:"N
100 PRINT "THIS PROGRAM ACCEPTS THREE FORMS OF INPUT"
110 PRINT
120 PRINT "   (1)  DIRECT INPUT AT THE TERMINAL"
130 PRINT "   (2)  READING A SET OF DATA STATEMENTS"
140 PRINT "   (3)  GENERATION OF A RANDOM PROBLEM"
150 PRINT
160 PRINT "THE INPUT FOR (1) AND (2) MUST BE IN THE FORM OF A"
170 PRINT "RELATIONSHIP MATRIX, USING THE STANDARD CHARACTERS"
180 PRINT "A,E,I,O,U,X , WITH CORRESPONDING NUMERICAL VALUES"
190 PRINT "OF 64,16,4,1,0,-128"
200 PRINT
210 PRINT "FOR (1) TYPE 'I' "
220 PRINT "FOR (2) TYPE 'R' "
230 PRINT "FOR (3) TYPE 'G' "
240 PRINT
250 INPUT "TYPE OF DATA INPUT REQUIRED"; P$
260 IF P$="I" GOTO 540
270 IF P$="R" GOTO 310
280 IF P$="G" GOTO 940
290 PRINT "INVALID OPTION. TRY AGAIN"
300 GOTO 250
310 FLAG%=0
320 LPRINT
330 LPRINT "RELATIONSHIP MATRIX IN THE FORM (I,J), FOR J ) I"
340 LPRINT
350 K=7
360 FOR I=1 TO N
370    LPRINT I " : ";
380    LPRINT TAB(K)
390    FOR J=I+1 TO N
400       READ P$
410       LPRINT P$ " ";
420       IF P$="U" THEN BEN%(I,J)=0  : GOTO 480
430       IF P$="O" THEN BEN%(I,J)=1  : GOTO 480
440       IF P$="I" THEN BEN%(I,J)=4  : GOTO 480
450       IF P$="E" THEN BEN%(I,J)=16 : GOTO 480
460       IF P$="A" THEN BEN%(I,J)=64 : GOTO 480
470       IF P$="X" THEN BEN%(I,J)=-128 : FLAG%=1
480    NEXT J
490    K=K+2
500    LPRINT
510 NEXT I
520 LPRINT
530 GOTO 790
540 PRINT
550 LPRINT
560 LPRINT "RELATIONSHIP MATRIX IN THE FORM (I,J), FOR J ) I"
570 LPRINT
580 FLAG%=0
590 K=7
600 FOR I=1 TO N
610    LPRINT I " : ";
620    LPRINT TAB(K)
630    FOR J=I+1 TO N
640       PRINT "REL CHART SCORE FOR FACILITIES " I " , " J " :";
```

```
650       INPUT P$
660       LPRINT P$ " ";
670       IF P$="U" THEN BEN%(I,J)=0 : GOTO 730
680       IF P$="O" THEN BEN%(I,J)=1 : GOTO 730
690       IF P$="I" THEN BEN%(I,J)=4 : GOTO 730
700       IF P$="E" THEN BEN%(I,J)=16: GOTO 730
710       IF P$="A" THEN BEN%(I,J)=64: GOTO 730
720       IF P$="X" THEN BEN%(I,J)=-128 : FLAG%=1
730    NEXT J
740    PRINT
750    K=K+2
760    LPRINT
770 NEXT I
780 LPRINT
790 IF FLAG%=0 GOTO 880
800 FOR I=1 TO N
810    FOR J=1 TO N
820       BEN%(I,J)=BEN%(I,J)+128
830    NEXT J
840 NEXT I
850 FOR I=1 TO N
860    BEN%(I,I)=0
870 NEXT I
880 FOR I=1 TO N
890    FOR J=I+1 TO N
900       BEN%(J,I)=BEN%(I,J)
910    NEXT J
920 NEXT I
930 GOTO 1130
940 RANDOMIZE
950 LPRINT "GENERATING RELATIONSHIP MATRIX RANDOMLY"
960 PRINT
970 PRINT "THIS ROUTINE WILL PROVIDE A RELATIONSHIP MATRIX ( NORMALLY"
980 PRINT "DISTRIBUTED ) FOR TESTING PURPOSES, USING THE METHOD OF BOX AND"
990 PRINT "MULLER. REQUIRED INPUT IS THE MEAN, MU, AND THE STANDARD"
1000 PRINT "DEVIATION, SIGMA, OF THE DESIRED DISTRIBUTION"
1010 PRINT
1020 INPUT "VALUE OF MU";MU%
1030 LPRINT "VALUE OF MU:" MU%
1040 INPUT "VALUE OF SIGMA";SIGMA%
1050 LPRINT "VALUE OF SIGMA:"SIGMA%
1060 FOR I=1 TO N
1070    FOR J=I+1 TO N
1080    BEN%(I,J)=INT(SQR(-2*LOG(RND))*COS(6.28319*RND)*SIGMA%+MU%)
1090    IF BEN%(I,J)<0 THEN BEN%(I,J)=0
1100    BEN%(J,I)=BEN%(I,J)
1110    NEXT J
1120 NEXT I
1130 LPRINT "RELATIONSHIP MATRIX IN NUMERICAL TERMS"
1140 LPRINT
1150 FOR I=1 TO N
1160    FOR J=1 TO N
1170       LPRINT BEN%(I,J);
1180    NEXT J
1190    LPRINT
1200 NEXT I
1210 LPRINT
1220 FOR I=1 TO N
1230    VALID%(I)=1
1240 NEXT I
1250 ORDER%(1)=1
1260 REM FIND BEST TETRAHEDRON INCLUDING THE EXTERIOR FACILITY
1270 MAX=-1
1280 FOR I=2 TO N-2
1290    FOR J=I+1 TO N-1
1300       FOR K=J+1 TO N
```

```
1310          SUM%=BEN%(I,1)+BEN%(J,1)+BEN%(K,1)+BEN%(I,J)+
                   BEN%(I,K)+BEN%(J,K)
1320          IF SUM%)MAX THEN MAX=SUM%:
                 ORDER%(2)=I:ORDER%(3)=J:ORDER%(4)=K
1330       NEXT K
1340    NEXT J
1350 NEXT I
1360 TOTBEN%=MAX
1370 FOR I=2 TO 4
1380    VALID%(ORDER%(I))=0
1390 NEXT I
1400 LPRINT
1410 LPRINT "BEST TETRAHEDRON:"
1420 LPRINT
1430 FOR I=1 TO 4
1440    LPRINT ORDER%(I);
1450 NEXT I
1460 LPRINT
1470 FOR I=1 TO 4
1480    X%=ORDER%(I)
1490    FOR J=1 TO 4
1500       Y%=ORDER%(J)
1510       IF J(I THEN TRIANG%(I,J)=Y%
          ELSE IF J)I THEN TRIANG%(I,J-1)=Y%:
          SOLUTION%(X%,Y%)=1 :SOLUTION%(Y%,X%)=1
1520    NEXT J
1530 NEXT I
1540 FOR I=1 TO N
1550    H%(I)=0
1560 NEXT I
1570 FOR I=2 TO 4
1580    CLASS%(1,I-1)=ORDER%(I)
1590    LAYER(1,I-1)=ORDER%(I)
1600    DIST%(ORDER%(I))=1
1610 REM DEFINE HASH FUNCTION FOR THE FUNDAMENTAL TRIANGLE
1620    H%(ORDER%(I))=I-2
1630 NEXT I
1640 PAIR%(1,1)=ORDER%(2)
1650 PAIR%(1,2)=ORDER%(3)
1660 PAIR%(2,1)=ORDER%(2)
1670 PAIR%(2,2)=ORDER%(4)
1680 PAIR%(3,1)=ORDER%(3)
1690 PAIR%(3,2)=ORDER%(4)
1700 DIST%(1)=0
1710 CARD%(1)=3
1720 MAXDIST=1
1730 FUNDTRI%(1)=1
1740 TRINO%=4
1750 VERTNO%=4
1760 WHILE VERTNO% ( N
1770    MAX=-1
1780    FOR I=2 TO N
1790    IF VALID%(I)=0 THEN 1900
1800       K=1
1810       WHILE K (= TRINO%
1820          SUM%=0
1830          FOR J=1 TO 3
1840             L=TRIANG%(K,J)
1850             SUM%=SUM%+BEN%(I,L)
1860          NEXT J
1870          IF SUM%) MAX THEN GOSUB 3640
1880          K=K+1
1890       WEND
1900    NEXT I
1910    FOR J=1 TO 3
1920       SOLUTION%(MAXX,TRIANG%(MAXTRI,J))=1
```

```
1930       SOLUTION%(TRIANG%(MAXTRI,J),MAXX)=1
1940    NEXT J
1950    GOSUB 3990
1960    TRINO%=TRINO%+1
1970    LPRINT
1980    LPRINT "INSERTING VERTEX" MAXX "IN TRIANGLE";
1990    FOR L=1 TO 3
2000       LPRINT TRIANG%(MAXTRI,L);
2010    NEXT L
2020    LPRINT
2030    TRIANG%(TRINO%,1)=TRIANG%(MAXTRI,1)
2040    TRIANG%(TRINO%,2)=TRIANG%(MAXTRI,2)
2050    TRIANG%(TRINO%,3)=MAXX
2060    TRINO%=TRINO%+1
2070    TRIANG%(TRINO%,1)=TRIANG%(MAXTRI,1)
2080    TRIANG%(TRINO%,2)=TRIANG%(MAXTRI,3)
2090    TRIANG%(TRINO%,3)=MAXX
2100    TRIANG%(MAXTRI,1)=MAXX
2110    VERTNO%=VERTNO%+1
2120    VALID%(MAXX)=0
2130    TOTBEN%=TOTBEN%+MAX
2140    CARD%(DIST%(MAXX))=CARD%(DIST%(MAXX))+1
2150    LAYER(DIST%(MAXX),CARD%(DIST%(MAXX)))=MAXX
2160 WEND
2170 LPRINT
2180 IF FLAG%=0 GOTO 2250
2190 TOTBEN%=TOTBEN%-(3*N-6)*128
2200 FOR I=1 TO N
2210    FOR J=I+1 TO N
2220       BEN%(I,J)=BEN%(I,J)-128
2230    NEXT J
2240 NEXT I
2250 LPRINT "TOTAL DELTAHEDRON ADJACENCY SCORE IS" TOTBEN%
2260 LPRINT
2270 REM WRITE OUT SOLUTION (INCIDENCE) MATRIX
2280 LPRINT "INCIDENCE MATRIX"
2290 LPRINT
2300 FOR I=1 TO N
2310    FOR J=1 TO N
2320       LPRINT SOLUTION%(I,J);
2330    NEXT J
2340    LPRINT
2350 NEXT I
2360 LPRINT
2370 REM SORT THE EDGES ACCORDING TO WEIGHT TO GET 3N-6 BOUND
2380 M=0
2390 FOR I=1 TO N
2400    FOR J=I+1 TO N
2410       M=M+1
2420       EDGES%(M)=BEN%(I,J)
2430    NEXT J
2440 NEXT I
2450 R%(1)=M\2
2460 T%=INT(LOG(M)/LOG(2))+1
2470 FOR I=1 TO T%-1
2480    R%(I+1)=INT(.75*R%(I))
2490 NEXT I
2500 FOR L=1 TO T%
2510    INC=R%(L)
2520    FOR I=1 TO M-INC
2530       IF EDGES%(I)(EDGES%(I+1) THEN SWAP EDGES%(I),EDGES%(I+INC)
2540    NEXT I
2550 NEXT L
2560 L=M-1
2570 WHILE L)0
2580    K=0
```

```
2590    FOR I=1 TO L
2600       IF EDGES%(I)<EDGES%(I+1) THEN SWAP EDGES%(I),EDGES%(I+1):K=I
2610    NEXT I
2620    L=K-1
2630 WEND
2640 LPRINT
2650 BOUND%=0
2660 FOR I=1 TO 3*N-6
2670    BOUND%=BOUND%+EDGES%(I)
2680 NEXT I
2690 LPRINT
2700 LPRINT "BEST POSSIBLE SOLUTION BOUND IS" BOUND%
2710 LPRINT
2720 LPRINT "DELTAHEDRON SOLUTION RATIO IS " TOTBEN%/BOUND%
2730 LPRINT
2740 LPRINT "DISTANCE CLASSES:"
2750 LPRINT
2760 FOR I=1 TO MAXDIST
2770    FOR J=1 TO CARD%(I)
2780       LPRINT LAYER(I,J);
2790    NEXT J
2800    LPRINT
2810 NEXT I
2820 LPRINT
2830 REM THE FOLLOWING STATEMENTS TRANSFER ALL RELEVANT DATA REQUIRED
2840 REM FOR THE LAYOUT PHASE TO DISK IN PREPARATION FOR READING BY
2850 REM THE SECOND PROGRAM, "PLAN". THIS IS NECESSARY IN ORDER THAT
2860 REM ENOUGH CORE IS AVAILABLE FOR RUNNING "PLAN" ON A 64K MICRO.
2870 REM THE FILES BEING READ TO IS FILE #1, TITLED "PROBDATA"
2880 OPEN "O",#1,"PROBDATA"
2890 PRINT #1,N
2900 PRINT #1,MAXDIST
2910 FOR I=1 TO N
2920    FOR J=1 TO N
2930       PRINT #1,SOLUTION%(I,J)
2940    NEXT J
2950 NEXT I
2960 FOR I=1 TO MAXDIST
2970    PRINT #1,CARD%(I)
2980 NEXT I
2990 FOR I=1 TO MAXDIST
3000    FOR J=1 TO 3
3010       PRINT #1,CLASS%(I,J)
3020    NEXT J
3030 NEXT I
3040 FOR I=1 TO N
3050    PRINT #1,BEFORE%(I)
3060 NEXT I
3070 FOR I=1 TO N
3080    PRINT #1,AND%(I)
3090 NEXT I
3100 FOR I=1 TO MAXDIST
3110    PRINT #1,KOUNT(I)
3120 NEXT I
3130 FOR I=1 TO N
3140    FOR J=1 TO AND%(I)
3150       PRINT #1,AFTER%(I,J)
3160    NEXT J
3170 NEXT I
3180 FOR I=1 TO MAXDIST
3190    PRINT #1,FUNDTRI%(I)
3200 NEXT I
3210 FOR I=1 TO MAXDIST
3220    PRINT #1,ORIGIN%(I)
3230 NEXT I
3240 FOR I=1 TO 3
```

```
3250    PRINT #1,NUM(I)
3260 NEXT I
3270 FOR I=1 TO 3
3280    FOR J=1 TO NUM(I)
3290       PRINT #1,BETW%(I,J)
3300    NEXT J
3310 NEXT I
3320 FOR I=1 TO N
3330    PRINT #1,H%(I)
3340 NEXT I
3350 FOR I=1 TO KOUNT(MAXDIST)
3360    PRINT #1,SAVED%(I)
3370 NEXT I
3380 CLOSE #1
3390 RUN "PLAN"
3400 END
3410 D%=DIST%(MAXX)
3420 IF BEFORE%(A%)=B% THEN SWAP A%,B%
3430 REM DETERMINE THE INDICES OF THE TWO VERTICES OF CLASS(D-1,*)
3440 REM TO WHICH B IS ADJACENT
3450 KK=1
3460 FOR K=1 TO 3
3470    IF SOLUTION%(B%,CLASS%(D%-1,K))=1 THEN
           BADJ%(KK)=K : KK=KK+1
3480 NEXT K
3490 REM SWAP INDICES IF NECESSARY
3500 IF ((BADJ%(1)=1) AND (BADJ%(2)=3)) THEN SWAP BADJ%(1),BADJ%(2)
3510 REM CHECK WHICH VERTEX MAXX IS ADJACENT TO
3520 REM IF THE FIRST, ORIENTATION IS (A,MAXX,B)
3530 REM IF THE SECOND, ORIENTATION IS (A,B,MAXX)
3540 IF SOLUTION%(MAXX,CLASS%(D%-1,BADJ%(1)))=1 THEN
        CLASS%(D%,1)=A%:CLASS%(D%,2)=MAXX:CLASS%(D%,3)=B%
    ELSE CLASS%(D%,1)=A%:CLASS%(D%,2)=B%:CLASS%(D%,3)=MAXX
3550 LPRINT
3560 LPRINT "ORIENTED CYCLE:"
3570 LPRINT
3580 LPRINT D% ":";
3590 FOR J=1 TO 3
3600    LPRINT CLASS%(D%,J);
3610 NEXT J
3620 LPRINT
3630 RETURN
3640 REM PREVENT CREATION OF A SECOND TRIANGLE IN THE DISTANCE CLASS
3650 A%=TRIANG%(K,1)
3660 B%=TRIANG%(K,2)
3670 C%=TRIANG%(K,3)
3680 IF DIST%(A%) <> DIST%(B%) THEN
        IF DIST%(B%)=DIST%(C%) THEN SWAP A%,C%
        ELSE SWAP B%,C%
3690 REM STOP CREATION OF TRIANGLES IN DISTANCE CLASS ONE WHICH ARE
3700 REM NOT OF AN APPROPRIATE TYPE ("CLASS 1")
3710 IF ((H%(A%)=-1) OR (H%(B%)=-1) OR (H%(C%)=-1)) THEN 3980
3720 REM PREVENT INSERTION IN ANY "CLASS 1" TRIANGLE OF DISTANCE CLASS
3730 REM ONE...THE ONLY ALLOWABLE SUCH TRIANGLE IS THE FUNDAMENTAL
3740 REM TRIANGLE, CONSTRUCTED AS TRIANG(1,*)
3750 IF DIST%(A%)=1 THEN
       IF ((DIST%(A%)=DIST%(B%)) AND (DIST%(B%)=DIST%(C%))) THEN
          IF K <> 1 THEN 3980 ELSE 3950
3760 REM PREVENT CREATION OF TRIANGLES IN OTHER DISTANCE CLASSES UNLESS
3770 REM THE FUNDAMENTAL TRIANGLE OF THAT CLASS IS YET TO BE DEFINED
3780 IF DIST%(A%) > 1 THEN
       IF ((DIST%(A%)=DIST%(C%)+1) AND (FUNDTRI%(DIST%(A%))=1)) GOTO 3980
3790 REM CONSTRAIN TO TWO DISTANCE CLASSES ONLY
3800 IF MAXDIST=2 THEN
       IF ((DIST%(A%)=DIST%(B%)) AND (DIST%(B%)=DIST%(C%))) GOTO 3980
3810 REM DETERMINE DISTANCE OF I TO EXTERIOR FACILITY IMPLIED BY
```

```
3820 REM ITS INSERTION IN TRIANGLE K
3830 MINIM=N
3840 FOR L=1 TO 3
3850    D%=DIST%(TRIANG%(K,L))
3860    IF D%(MINIM THEN MINIM=D%
3870 NEXT L
3880 DIST%(I)=MINIM+1
3890 REM NOW DETERMINE WHETHER ADDING I TO DISTANCE CLASS DIST(I)
3900 REM IS VALID IN TERMS OF THE CONSTRAINED APPROACH...I.E. DOES
3910 REM THE DISTANCE CLASS REMAIN CONNECTED?
3920 IF CARD%(DIST%(I))=0 THEN 3950
3930 IF ((DIST%(A%)=DIST%(B%)) AND (DIST%(B%)=DIST%(C%))) THEN
       IF CARD%(DIST%(I)) ) 0 THEN 3980
3940 REM ACCEPT TRIANG(K) AS TRIANGLE FOR INSERTION
3950 MAX=SUM%
3960 MAXTRI%=K
3970 MAXX=I
3980 RETURN
3990 REM ELEMENTS OF INSERTION TRIANGLE ARE TRIANG(MAXTRI,*)
4000 A%=TRIANG%(MAXTRI,1)
4010 B%=TRIANG%(MAXTRI,2)
4020 C%=TRIANG%(MAXTRI,3)
4030 IF DIST%(A%) () DIST%(B%) THEN
       IF DIST%(B%)=DIST%(C%) THEN SWAP A%,C%
         ELSE SWAP B%,C%
4040 REM NOW HAVE ONE OF THE FOLLOWING CASES
4050 REM    (1) DIST(A)=DIST(B)=DIST(C)+1
4060 REM    (2) DIST(A)=DIST(B)=DIST(C)-1
4070 REM    (3) DIST(A)=DIST(B)=DIST(C)
4080 DA%=DIST%(A%)
4090 DB%=DIST%(B%)
4100 DC%=DIST%(C%)
4110 IF DA%=DC%+1 THEN 4180
4120 IF DA%=DC%-1 THEN 4690
4130 REM OTHERWISE, WE HAVE CASE (3)
4140 DIST%(MAXX)=DA%+1
4150 MAXDIST=MAXDIST+1
4160 ORIGIN%(DIST%(MAXX))=MAXX
4170 GOTO 4740
4180 REM CASE (1)
4190 IF C%=1 GOTO 4380
4200 FUNDTRI%(DA%)=1
4210 DIST%(MAXX)=DA%
4220 REM DEFINE AND ORIENT CLASS(DA,*)
4230 GOSUB 3410
4240 IF BEFORE%(A%)=B% THEN Q%=B%:FUND%(DA%,1)=B%:
                            FUND%(DA%,2)=A%: GOTO 4280
4250 Q%=A%
4260 FUND%(DA%,1)=A%
4270 FUND%(DA%,2)=B%
4280 IF BEFORE%(Q%)=0 GOTO 4740
4290 SAVEQ%=Q%
4300 WHILE Q% () ORIGIN%(DA%)
4310    AFTER%(Q%,1)=BEFORE%(Q%)
4320    SAVEQ%=Q%
4330    KOUNT(DA%)=KOUNT(DA%)+1
4340    SAVED%(KOUNT(DA%))=BEFORE%(Q%)
4350    Q%=BEFORE%(Q%)
4360 WEND
4370 GOTO 4740
4380 IF H%(A%) () -2 THEN 4420
4390 FOR KK=1 TO 3
4400    IF STORE%(KK,1)=A% THEN L=H%(STORE%(KK,2))+H%(STORE%(KK,3)):GOTO 4560
4410 NEXT KK
4420 IF H%(B%) () -2 THEN 4460
4430 FOR KK=1 TO 3
```

```
4440    IF STORE%(KK,1)=B% THEN L=H%(STORE%(KK,2))+H%(STORE%(KK,3)):GOTO 4560
4450 NEXT KK
4460 L=H%(A%)+H%(B%)
4470 IF NUM(L) ) 0 GOTO 4550
4480 STORE%(L,1)=MAXX
4490 STORE%(L,2)=A%
4500 STORE%(L,3)=B%
4510 H%(MAXX)=-2
4520 NUM(L)=1
4530 BETW%(L,1)=MAXX
4540 GOTO 4670
4550 IF NUM(L) ) 1 GOTO 4620
4560 FOR LL=1 TO 2
4570    IF A%=PAIR%(L,LL) THEN
           HMAXX%(L)=H%(PAIR%(L,LL MOD 2 +1)) : GOTO 4620
4580 NEXT LL
4590 FOR LL=1 TO 2
4600    IF B%=PAIR%(L,LL) THEN
           HMAXX%(L)=H%(PAIR%(L,LL MOD 2 +1)) : GOTO 4620
4610 NEXT LL
4620 H%(MAXX)=HMAXX%(L)
4630 H%(STORE%(L,1))=-1
4640 STORE%(L,1)=MAXX
4650 NUM(L)=NUM(L)+1
4660 BETW%(L,NUM(L))=MAXX
4670 DIST%(MAXX)=1
4680 GOTO 4740
4690 REM CASE (2)
4700 ANO%(C%)=ANO%(C%)+1
4710 AFTER%(C%,ANO%(C%))=MAXX
4720 BEFORE%(MAXX)=C%
4730 DIST%(MAXX)=DC%
4740 RETURN
4750 DATA I,I,U,U,A,A,U,U,I,E,A,U
4760 DATA I,U,X,X,U,U,U,U,D,D,U
4770 DATA U,U,U,U,I,U,U,U,U,U
4780 DATA A,I,E,E,U,E,U,I,I
4790 DATA U,I,X,U,E,I,I,I
4800 DATA I,X,U,U,I,E,I
4810 DATA E,I,E,I,I,I
4820 DATA E,U,A,U,U
4830 DATA U,E,I,U
4840 DATA U,D,D
4850 DATA U,D
4860 DATA U
```

```
10 REM OUTPUT BLOCK PLAN USING INPUT FROM "DELTAH"
20 DEFINT I-N
30 DIM SOLUTION%(30,30),CHAR$(30)
40 DIM CARD%(10),TAILLENG%(3),PLAN%(60,60)
50 DIM CLASS%(10,3),BETW%(3,10),H%(30)
60 DIM FUNDTRI%(10),ORIGIN%(10),SAVED%(10)
70 DIM ANO%(30),BEFORE%(30),AFTER%(30,10),KOUNT(10),NUM(30)
80 OPEN "I",#1,"PROBDATA"
90 INPUT #1,N
100 INPUT #1,MAXDIST
110 FOR I=1 TO N
120    FOR J=1 TO N
130       INPUT #1,SOLUTION%(I,J)
140    NEXT J
150 NEXT I
160 FOR I=1 TO MAXDIST
170    INPUT #1,CARD%(I)
180 NEXT I
190 FOR I=1 TO MAXDIST
200    FOR J=1 TO 3
210       INPUT #1,CLASS%(I,J)
220    NEXT J
230 NEXT I
240 FOR I=1 TO N
250    INPUT #1,BEFORE%(I)
260 NEXT I
270 FOR I=1 TO N
280    INPUT #1,ANO%(I)
290 NEXT I
300 FOR I=1 TO MAXDIST
310    INPUT #1,KOUNT(I)
320 NEXT I
330 FOR I=1 TO N
340    FOR J=1 TO ANO%(I)
350       INPUT #1,AFTER%(I,J)
360    NEXT J
370 NEXT I
380 FOR I=1 TO MAXDIST
390    INPUT #1,FUNDTRI%(I)
400 NEXT I
410 FOR I=1 TO MAXDIST
420    INPUT #1,ORIGIN%(I)
430 NEXT I
440 FOR I=1 TO 3
450    INPUT #1,NUM(I)
460 NEXT I
470 FOR I=1 TO 3
480    FOR J=1 TO NUM(I)
490       INPUT #1,BETW%(I,J)
500    NEXT J
510 NEXT I
520 FOR I=1 TO N
530    INPUT #1,H%(I)
540 NEXT I
541 FOR I=1 TO KOUNT(MAXDIST)
542    INPUT #1,SAVED%(I)
543 NEXT I
550 CLOSE #1
560 FOR I=1 TO N
570    READ CHAR$(I)
580 NEXT I
590 DATA H,I,(,0,V,X,\,$,*,+,-,/,&,=,@,),#,%,A,B,C,D,E,F,G,J,K,L,M,N,P
600 INPUT "RATIO OF BUILDING LENGTH TO WIDTH ()=1)"; RATIO
610 INPUT "HORIZONTAL SIZE OF PLAN MATRIX DESIRED"; WID%
```

```
620 REM ARTIFICIALLY CREATE SECOND DISTANCE CLASS WITH ZERO ELEMENTS
630 REM FOR THE CASE SUCH THAT CARD(2)=0
640 IF CARD%(2)=0 THEN MAXDIST=2
650 IF CARD%(MAXDIST) ) 1 GOTO 700
660 TOP%=CLASS%(MAXDIST-1,1)
670 LEFT=CLASS%(MAXDIST-1,2)
680 RIGHT%=CLASS%(MAXDIST-1,3)
690 GOTO 1380
700 IF FUNDTRI%(MAXDIST)=1 GOTO 970
710 IF AND%(ORIGIN%(MAXDIST)) ) 1 GOTO 830
720 K=AFTER%(ORIGIN%(MAXDIST),1)
730 FOR L=1 TO 3
740    IF SOLUTION%(K,CLASS%(MAXDIST-1,L))()1 GOTO 760
750 NEXT L
760 TOP%=CLASS%(MAXDIST-1,L)
770 IF L=3 THEN L=1 ELSE L=L+1
780 LEFT=CLASS%(MAXDIST-1,L)
790 L=L+1
800 IF L=4 THEN L=1
810 RIGHT%=CLASS%(MAXDIST-1,L)
820 GOTO 1380
830 IF AND%(ORIGIN%(MAXDIST)) ) 2 GOTO 660
840 K=AFTER%(ORIGIN%(MAXDIST).1)
850 L=AFTER%(ORIGIN%(MAXDIST),2)
860 FOR I=1 TO 3
870    IF ((SOLUTION%(K,CLASS%(MAXDIST-1,I))=1) AND
            (SOLUTION%(L,CLASS%(MAXDIST-1,I)))=1) GOTO 890
880 NEXT I
890 IF I=3 THEN I=1 ELSE I=I+1
900 TOP%=CLASS%(MAXDIST-1,I)
910 IF I=3 THEN I=1 ELSE I=I+1
920 LEFT=CLASS%(MAXDIST-1,I)
930 I=I+1
940 IF I=4 THEN I=1
950 RIGHT%=CLASS%(MAXDIST-1,I)
960 GOTO 1380
970 REM FIRSTLY DETERMINE INNER SANCTUM POSITIONS
980 FOR K=1 TO 3
990 IF ((BEFORE%(CLASS%(MAXDIST,K))=0) AND
       (CLASS%(MAXDIST,K) () ORIGIN%(MAXDIST))) GOTO 1010
1000 NEXT K
1010 INRIGHT=CLASS%(MAXDIST,K)
1020 L=K MOD 3 +1
1030 LL=L MOD 3 +1
1040 IF BEFORE%(CLASS%(MAXDIST,L))=CLASS%(MAXDIST,LL)
       THEN C%=L ELSE C%=LL
1050 INLEFT=CLASS%(MAXDIST,C%)
1060 FOR I=1 TO 3
1070    IF ((I () K) AND (I () C%)) THEN KL=I
1080 NEXT I
1090 INTOP=CLASS%(MAXDIST,KL)
1100 REM NOW DEFINE CLASS(1) POSITIONS RELATIVE TO THE INNER SANCTUM
1110 FOR I=1 TO 3
1120    IF SOLUTION%(INRIGHT,CLASS%(MAXDIST-1,I))=1
          THEN L=I : GOTO 1140
1130 NEXT I
1140 RIGHT%=CLASS%(MAXDIST-1,L)
1150 FOR I=1 TO 3
1160    IF I=L GOTO 1180
1170    IF SOLUTION%(CLASS%(MAXDIST-1,I),INLEFT)=1 THEN LL=I:GOTO 1190
1180 NEXT I
1190 LEFT=CLASS%(MAXDIST-1,LL)
1200 FOR I=1 TO 3
1210    IF ((I () L) AND (I () LL)) THEN KL=I
1220 NEXT I
1230 TOP%=CLASS%(MAXDIST-1.KL)
```

```
1380 REM CALCULATE THE BOUNDARIES OF THE DISTANCE CLASSES IN THE PLAN
1390 REM SO THAT THE TOTAL AREA ALLOCATED TO EACH CLASS IS PROPORTIONAL
1400 REM TO THE CARDINALITY OF THE DISTANCE CLASS
1410 REM ALSO ASSUME CONGRUENCE
1420 REM WIDTH1 IS THE HORIZONTAL WIDTH OF CLASS 1
1430 REM IF THERE IS NO INNER SANCTUM, HAVE TO SPECIFY WIDTHS DIFFERENTLY
1440 IF CARD%(MAXDIST) <> 0 GOTO 1530
1450 WIDTH1%=WID%*(NUM(H%(TOP%)+H%(LEFT))+1)\(CARD%(1)-1)
1460 WIDTH1R%=WID%*RATIO/CARD%(1)
1470 WIDTH2R%=0
1480 TOTDEPTH%=WID%*RATIO
1490 W12%=WIDTH1%
1500 W12R%=WIDTH1R%
1510 GOTO 1630
1520 REM WIDTH2 IS THE HORIZONTAL WIDTH OF CLASS(MAXDIST)
1530 WIDTH2%=SQR(CARD%(MAXDIST)*WID%*WID%/N)
1540 REM APPROXIMATE WIDTH2 BY AN EVEN INTEGER
1550 IF WIDTH2% MOD 2 =1 THEN WIDTH2%=WIDTH2%-1
1560 WIDTH1%=(WID%-WIDTH2%)\2
1570 WIDTH1R%=WIDTH1%*RATIO
1580 WIDTH2R%=WIDTH2%*RATIO
1590 TOTDEPTH%=2*WIDTH1R%+WIDTH2R%
1600 REM HENCE PLAN MATRIX HAS DIMENSION WIDTH*TOTDEPTH
1610 W12%=WIDTH1%+WIDTH2%
1620 W12R%=WIDTH1R%+WIDTH2R%
1630 REM FILL IN TOP
1640 FOR I=1 TO WIDTH1R%-1
1650    FOR J=1 TO WID%
1660       PLAN%(I,J)=TOP%
1670    NEXT J
1680 NEXT I
1690 REM FILL IN LEFT
1700 FOR I=WIDTH1R% TO TOTDEPTH%
1710    FOR J=1 TO WIDTH1%-1
1720       PLAN%(I,J)=LEFT
1730    NEXT J
1740 NEXT I
1750 REM FILL IN RIGHT
1760 FOR I=WIDTH1R% TO TOTDEPTH%
1770    FOR J=W12% TO WID%
1780       PLAN%(I,J)=RIGHT%
1790    NEXT J
1800 NEXT I
1810 IF CARD%(MAXDIST)=0 GOTO 1870
1820 FOR I=W12R% TO TOTDEPTH%
1830    FOR J=WIDTH1% TO W12%-1
1840       PLAN%(I,J)=RIGHT%
1850    NEXT J
1860 NEXT I
1870 REM FILL IN BETWEEN TOP AND LEFT, IF ANYTHING EXISTS
1880 HTL%=H%(TOP%)+H%(LEFT)
1890 IF NUM(HTL%)=0 GOTO 2270
1900 HW%=(WIDTH1%-1)\(NUM(HTL%)+1)
1910 DW%=(TOTDEPTH%-WIDTH1R%)\(NUM(HTL%)+1)
1920 FINX%=HW%*NUM(HTL%)
1930 FINY%=WIDTH1R%+DW%*NUM(HTL%)-1
1940 FOR I=WIDTH1R% TO FINY%
1950    FOR J=1 TO FINX%
1960       PLAN%(I,J)=BETW%(HTL%,1)
1970    NEXT J
1980 NEXT I
1990 IF NUM(HTL%)=1 GOTO 2270
2000 REM DETERMINE ORIENTATION OF THE TAIL, W.R.T. TOP,LEFT
2010 IF SOLUTION%(BETW%(HTL%,2),TOP%)=1 GOTO 2150
2020 REM TAIL IS ADJACENT TO LEFT
2030 STARTY%=WIDTH1R%+DW%
```

```
2050 FOR K=2 TO NUM(HTL%)
2060    FOR I=STARTY% TO FINY%
2070       FOR J=1 TO FINX%
2080          PLAN%(I,J)=BETW%(HTL%,K)
2090       NEXT J
2100    NEXT I
2110    STARTY%=STARTY%+DW%
2130 NEXT K
2140 GOTO 2270
2150 REM TAIL IS ADJACENT TO TOP
2160 FINX%=FINX%-HW%
2170 STARTY%=FINY%-DW%
2180 FOR K=2 TO NUM(HTL%)
2190    FOR I=STARTY% TO WIDTH1R% STEP -1
2200       FOR J=1 TO FINX%
2210          PLAN%(I,J)=BETW%(HTL%,K)
2220       NEXT J
2230    NEXT I
2240    STARTY%=STARTY%-DW%
2250    FINX%=FINX%-HW%
2260 NEXT K
2270 REM FILL IN BETWEEN TOP AND RIGHT, IF ANYTHING EXISTS
2280 HTR%=H%(TOP%)+H%(RIGHT%)
2290 IF NUM(HTR%)=0 GOTO 2710
2300 IF CARD%(MAXDIST) <> 0 GOTO 2340
2310 HW%=(WID%-WIDTH1%)\(NUM(HTR%)+1)
2320 DW%=(TOTDEPTH%-WIDTH1R%)\(NUM(HTR%)+1+NUM(H%(LEFT)+H%(RIGHT%)))
2330 GOTO 2360
2340 HW%=(WID%-W12%)\(NUM(HTR%)+1)
2350 DW%=(TOTDEPTH%-WIDTH1R%)\(NUM(HTR%)+1)
2360 STARTX%=WID%-HW%*NUM(HTR%)
2370 FINY%=WIDTH1R%+DW%*NUM(HTR%)-1
2380 FOR I=WIDTH1R% TO FINY%
2390    FOR J=STARTX% TO WID%
2400       PLAN%(I,J)=BETW%(HTR%,1)
2410    NEXT J
2420 NEXT I
2430 IF NUM(HTR%)=1 GOTO 2710
2440 REM DETERMINE ORIENTATION OF TAIL W.R.T. TOP,RIGHT
2450 IF SOLUTION%(BETW%(HTR%,2),TOP%)=1 GOTO 2590
2460 REM TAIL IS ADJACENT TO RIGHT
2470 STARTY%=WIDTH1R%+DW%
2490 FOR K=2 TO NUM%(HTR%)
2500    FOR I=STARTY% TO FINY%
2510       FOR J=STARTX% TO WID%
2520          PLAN%(I,J)=BETW%(HTR%,K)
2530       NEXT J
2540    NEXT I
2550    STARTY%=STARTY%+DW%
2570 NEXT K
2580 GOTO 2710
2590 REM TAIL IS ADJACENT TO TOP
2600 STARTY%=FINY%-DW%
2610 STARTX%=STARTX%+HW%
2620 FOR K=2 TO NUM(HTR%)
2630    FOR I=STARTY% TO WIDTH1R% STEP -1
2640       FOR J=STARTX% TO WID%
2650          PLAN%(I,J)=BETW%(HTR%,K)
2660       NEXT J
2670    NEXT I
2680    STARTY%=STARTY%-DW%
2690    STARTX%=STARTX%+HW%
2700 NEXT K
2710 REM FILL IN BETWEEN LEFT AND RIGHT, IF ANYTHING EXISTS
2720 HLR%=H%(LEFT)+H%(RIGHT%)
2730 IF NUM(HLR%)=0 GOTO 3140
```

```
2740 IF CARD%(MAXDIST) <> 0 GOTO 2770
2750 HW%=(WID%-WIDTH1%)\(NUM(HLR%)+1)
2760 GOTO 2790
2770 HW%=(W12%-WIDTH1%)\(NUM(HLR%)+1)
2780 DW%=(TOTDEPTH%-W12R%)\(NUM(HLR%)+1)
2790 FINX%=WIDTH1%+HW%*NUM(HLR%)-1
2800 STARTY%=TOTDEPTH%-DW%*NUM(HLR%)
2810 FOR I=STARTY% TO TOTDEPTH%
2820    FOR J=WIDTH1% TO FINX%
2830       PLAN%(I,J)=BETW%(HLR%,1)
2840    NEXT J
2850 NEXT I
2860 IF NUM(HLR%)=1 GOTO 3140
2870 REM DETERMINE ORIENTATION W.R.T. LEFT,RIGHT
2880 IF SOLUTION%(BETW%(HTR%,2),RIGHT%)=1 GOTO 3020
2890 REM TAIL IS ADJACENT TO LEFT
2900 FINX%=FINX%-HW%
2910 STARTY%=STARTY%+DW%
2920 FOR K=2 TO NUM(HLR%)
2930    FOR I=STARTY% TO TOTDEPTH%
2940       FOR J=WIDTH1% TO FINX%
2950          PLAN%(I,J)=BETW%(HLR%,K)
2960       NEXT J
2970    NEXT I
2980    STARTY%=STARTY%+DW%
2990    FINX%=FINX%-HW%
3000 NEXT K
3010 GOTO 3140
3020 REM TAIL IS ADJACENT TO RIGHT
3030 STARTX%=FINX%-HW%*(NUM(HLR%)-1)+1
3050 FOR K=2 TO NUM(HLR%)
3060    FOR I=STARTY% TO TOTDEPTH%
3070       FOR J=STARTX% TO FINX%
3080          PLAN%(I,J)=BETW%(HLR%,K)
3090       NEXT J
3100    NEXT I
3110    STARTX%=STARTX%+HW%
3130 NEXT K
3140 REM FILL IN INNER SANCTUM
3150 IF CARD%(MAXDIST)=0 GOTO 5350
3160 OM%=ORIGIN%(MAXDIST)
3170 IF CARD%(MAXDIST) > 1 GOTO 3240
3180 FOR I=WIDTH1R% TO W12R%-1
3190    FOR J=WIDTH1% TO W12%-1
3200       PLAN%(I,J)=OM%
3210    NEXT J
3220 NEXT I
3230 GOTO 5350
3240 IF FUNDTRI%(MAXDIST)=1 GOTO 4480
3250 IF AND%(OM%) > 1 GOTO 3550
3260 REM OTHERWISE, HAVE ONLY A PATH
3270 REM ESTABLISH PATH CARDINALITY
3280 PATHL%=1
3290 K=AFTER%(OM%,1)
3300 WHILE K <> 0
3310    PATHL%=PATHL%+1
3320    SAVEK%=K
3330    K=AFTER%(K,1)
3340 WEND
3350 DW%=WIDTH2R%\PATHL%
3360 K=OM%
3370 STARTY%=WIDTH1R%
3380 WHILE K <> SAVEK%
3390    FINY%=STARTY%+DW%-1
3400    FOR I=STARTY% TO FINY%
3410       FOR J=WIDTH1% TO W12%-1
```

```
3420          PLAN%(I,J)=K
3430       NEXT J
3440    NEXT I
3450    STARTY%=STARTY%+DW%
3460    K=AFTER%(K,1)
3470 WEND
3480 REM FILL IN SAVEK, IN CASE OF NON-INTEGRAL DIVISION
3490 FOR I=STARTY% TO W12R%-1
3500    FOR J=WIDTH1% TO W12%-1
3510       PLAN%(I,J)=SAVEK%
3520    NEXT J
3530 NEXT I
3540 GOTO 5350
3550 IF ANO%(OM%) ) 2 GOTO 3920
3560 REM CHECK WHICH TAIL IS ADJACENT TO LEFT AND RIGHT
3570 REM (ONE OF THEM MUST BE, BY DEFINITION OF TOP, LEFT, RIGHT)
3580 IF ((SOLUTION%(LEFT,AFTER%(OM%),1))=1) AND
       (SOLUTION%(RIGHT%,AFTER%(OM%,1))=1)) GOTO 3610
3590 LL=2
3600 GOTO 3620
3610 LL=1
3620 ML=LL MOD 2 +1
3630 REM TAIL LL IS ADJACENT TO LEFT AND RIGHT
3640 REM TAIL ML IS ADJACENT TO LEFT,TOP OR RIGHT,TOP
3650 REM DETERMINE LENGTH OF TAIL ML
3660 K=AFTER%(OM%,ML)
3670 LENGTH=0
3680 WHILE K () 0
3690    LENGTH=LENGTH+1
3700    K=AFTER%(K,1)
3710 WEND
3720 REM CALCULATE LENGTH OF TAIL LL, LLL
3730 LLL=CARD%(MAXDIST)-LENGTH-1
3740 DW%=WIDTH2R%\CARD%(MAXDIST)
3750 HW%=WIDTH2%\CARD%(MAXDIST)
3760 REM FILL IN ORIGIN(MAXDIST)
3770 FINY%=(LENGTH+1)*DW%-1
3780 FOR I=WIDTH1R% TO FINY%
3790    FOR J=WIDTH1% TO W12%-1
3800       PLAN%(I,J)=OM%
3810    NEXT J
3820 NEXT I
3830 REM FILL IN TAIL ML
3840 X%=ML
3850 IF SOLUTION%(AFTER%(OM%,ML),LEFT)=1 GOTO 3880
3860 GOSUB 5690
3870 GOTO 3890
3880 GOSUB 5540
3890 REM FILL IN TAIL LL
3900 GOSUB 5990
3910 GOTO 5350
3920 REM ANO(OM)=3
3930 REM CHECK WHICH TAIL IS ADJACENT TO LEFT AND RIGHT
3940 FOR I=1 TO 3
3950    IF ((SOLUTION%(LEFT,AFTER%(OM%,I))=1) AND
          (SOLUTION%(RIGHT%,AFTER%(OM%,I))=1)) GOTO 3970
3960 NEXT I
3970 LL=I
3980 REM CHECK WHICH TAIL IS ADJACENT TO LEFT,TOP
3990 FOR I=1 TO 3
4000 IF I=LL GOTO 4020
4010    IF (( SOLUTION%(LEFT,AFTER%(OM%,I))=1) AND
             (SOLUTION%(TOP%,AFTER%(OM%,I))=1)) GOTO 4030
4020 NEXT I
4030 KL=I
4040 REM THE OTHER TAIL IS ADJACENT TO RIGHT,TOP
```

```
4050 FOR I=1 TO 3
4060    IF ((I () LL) AND (I () KL)) THEN ML=I : GOTO 4080
4070 NEXT I
4080 REM TAIL LL IS ADJACENT TO LEFT,RIGHT
4090 REM TAIL KL IS ADJACENT TO TOP,LEFT
4100 REM TAIL ML IS ADJACENT TO TOP, RIGHT
4110 REM DETERMINE THE LENGTH OF TAILS KL,ML,LL
4120 K=AFTER%(OM%,KL)
4130 LKL=0
4140 WHILE K () 0
4150    LKL=LKL+1
4160    K=AFTER%(K,1)
4170 WEND
4180 LML=0
4190 K=AFTER%(OM%,ML)
4200 WHILE K () 0
4210    LML=LML+1
4220    K=AFTER%(K,1)
4230 WEND
4240 LLL=CARD%(MAXDIST)-LKL-LML-1
4250 REM DETERMINE HORIZONTAL AND VERTICAL DIVISIONS
4260 IF LKL)LML THEN MAX=LKL ELSE MAX=LML
4270 DW%=WIDTH2R%\(MAX+LLL+1)
4280 HW%=WIDTH2%\(LKL+LML+1)
4290 REM FILL IN TOP
4300 FINY%=(MAX+1)*DW%-1
4310 FOR I=WIDTH1R% TO FINY%
4320    FOR J=WIDTH1% TO W12%-1
4330       PLAN%(I,J)=OM%
4340    NEXT J
4350 NEXT I
4360 REM FILL IN TOP,LEFT
4370 LENGTH=LKL
4380 X%=KL
4390 GOSUB 5540
4400 REM FILL IN TOP,RIGHT
4410 LENGTH=LML
4420 X%=ML
4430 GOSUB 5690
4440 REM FILL IN TAIL LL
4450 LENGTH=MAX
4460 GOSUB 5990
4470 GOTO 5350
4480 REM NOW HAVE CASE WHERE FUNDTRI(MAXDIST)=1
4490 REM HAVE ALREADY DETERMINED POSITIONS OF THE TRIANGLE IN THE INNER
4500 REM SANCTUM (INTOP,INLEFT,INRIGHT)
4510 REM CHECK LENGTH OF TAILS ( AND KOUNT(MAXDIST))
4520 FOR I=1 TO ANO%(OM%)
4530 K=AFTER%(OM%,I)
4540 IF ((SOLUTION%(K,LEFT)=1) AND (SOLUTION%(K,TOP%)=1))
       THEN LTTAIL=I
     ELSE IF ((SOLUTION%(K,RIGHT%)=1) AND (SOLUTION%(K,TOP%)=1))
        THEN RTTAILX=I  ELSE 4610
4550    KK=0
4560    WHILE K () 0
4570       KK=KK+1
4580       K=AFTER%(K,1)
4590    WEND
4600    TAILLENG%(I)=KK
4610 NEXT I
4620 REM DETERMINE LENGTH OF TAIL FROM INLEFT, IF IT EXISTS
4630 K=AFTER%(INLEFT,1)
4640 KK=0
4650 WHILE K () 0
4660    KK=KK+1
4670    K=AFTER%(K,1)
```

```
4680 WEND
4690 REM DETERMINE DW,HW
4700 IF TAILLENG%(LTTAIL) ) TAILLENG%(RTTAIL%) THEN
       MAX=TAILLENG%(LTTAIL) ELSE MAX=TAILLENG%(RTTAIL%)
4710 DW%=WIDTH2%\(MAX+KOUNT(MAXDIST)+2+KK)
4720 HW%=WIDTH2%\(TAILLENG%(LTTAIL)+TAILLENG%(RTTAIL%)+1)
4730 REM FILL IN OM
4740 FINY%=(MAX+1)*DW%-1+WIDTH1R%
4750 FOR I=WIDTH1R% TO FINY%
4760    FOR J=WIDTH1% TO W12%-1
4770       PLAN%(I,J)=OM%
4780    NEXT J
4790 NEXT I
4800 REM FILL IN TAILS LTTAIL AND RTTAIL
4810 IF MAX=0 GOTO 4900
4820 IF LTTAIL=0 GOTO 4860
4830 LENGTH=TAILLENG%(LTTAIL)
4840 X%=LTTAIL
4850 GOSUB 5540
4860 IF RTTAIL% =0 GOTO 4900
4870 LENGTH=TAILLENG%(RTTAIL%)
4880 X%=RTTAIL%
4890 GOSUB 5690
4900 REM FILL IN BETWEEM OM AND INTOP, IF ANYTHING EXISTS
4910 STARTY%=(MAX+1)*DW%+WIDTH1R%
4920 IF KOUNT(MAXDIST)=0 GOTO 5060
4970 FOR K=KOUNT(MAXDIST)-1 TO 1 STEP -1
4980    FOR I=STARTY% TO STARTY%+DW%-1
4990       FOR J=WIDTH1% TO W12%-1
5000          PLAN%(I,J)=SAVED%(K)
5010       NEXT J
5020    NEXT I
5040    STARTY%=STARTY%+DW%
5050 NEXT K
5060 REM FILL IN INTOP
5070 IF INTOP=OM% GOTO 5140
5080 FOR I=STARTY% TO STARTY%+DW%
5090    FOR J=WIDTH1% TO W12%-1
5100       PLAN%(I,J)=INTOP
5110    NEXT J
5120 NEXT I
5130 STARTY%=STARTY%+DW%
5140 REM FILL IN INLEFT,INRIGHT TO THE END (W12R-1)
5150 HW%=WIDTH2%\(KK+2)
5160 FINX%=(KK+1)*HW%-1+WIDTH1%
5170 FOR I=STARTY% TO W12R%-1
5180    FOR J=WIDTH1% TO FINX%
5190       PLAN%(I,J)=INLEFT
5200    NEXT J
5210 NEXT I
5220 STARTX%=FINX%+1
5230 FOR I=STARTY% TO W12R%-1
5240    FOR J=STARTX% TO W12%-1
5250       PLAN%(I,J)=INRIGHT
5260    NEXT J
5270 NEXT I
5280 REM FILL IN TAIL EMANATING FROM INLEFT
5290 IF KK=0 GOTO 5350
5300 HW%=(FINX%-WIDTH1%)\(KK+1)
5310 LENGTH=KK
5320 X%=1
5330 OM%=INLEFT
5340 GOSUB 5840
5350 REM WRITE OUT PLAN MATRIX
5360 LPRINT
5370 LPRINT CHR$(27) CHR$(56)
```

```
5380 LPRINT CHR$(29) CHR$(31)
5390 FOR I=1 TO TOTDEPTH%
5400    FOR J=1 TO WID%
5410       LPRINT CHAR$(PLAN%(I,J));
5420    NEXT J
5430    LPRINT
5440 NEXT I
5450 LPRINT CHR$(27) CHR$(54)
5460 LPRINT CHR$(30)
5470 LPRINT
5480 LPRINT "CHARACTER SET CODES:"
5490 LPRINT
5500 FOR I=2 TO N
5510    LPRINT "FACILITY" I "IS REPRESENTED BY  " CHAR$(I)
5520 NEXT I
5530 END
5540 REM TAIL BETWEEN TOP AND LEFT
5550 FINX%=WIDTH1% + LENGTH*HW%-1
5560 STARTY%=WIDTH1R% +LENGTH*DW%-1
5570 K=AFTER%(OM%,X%)
5580 FOR L=1 TO LENGTH
5590    FOR I=STARTY% TO WIDTH1R% STEP -1
5600       FOR J=WIDTH1% TO FINX%
5610          PLAN%(I,J)=K
5620       NEXT J
5630    NEXT I
5640    FINX%=FINX%-HW%
5650    STARTY%=STARTY%-DW%
5660    K=AFTER%(K,1)
5670 NEXT L
5680 RETURN
5690 REM TAIL BETWEEN RIGHT AND TOP
5700 STARTX%=W12%-LENGTH*HW%
5710 STARTY%=WIDTH1R%+LENGTH*DW%-1
5720 K=AFTER%(OM%,X%)
5730 FOR L=1 TO LENGTH
5740    FOR I=STARTY% TO WIDTH1R% STEP -1
5750       FOR J=STARTX% TO W12%-1
5760          PLAN%(I,J)=K
5770       NEXT J
5780    NEXT I
5790    STARTX%=STARTX%+HW%
5800    STARTY%=STARTY%-DW%
5810    K=AFTER%(K,1)
5820 NEXT L
5830 RETURN
5840 REM TAIL BETWEEN LEFT AND RIGHT
5850 STARTY%=W12R%-LENGTH*DW%
5860 FINX%=WIDTH1%+LENGTH*HW%-1
5870 K=AFTER%(OM%,X%)
5880 FOR L=1 TO LENGTH
5890    FOR I=STARTY% TO W12R%-1
5900       FOR J=WIDTH1% TO FINX%
5910          PLAN%(I,J)=K
5920       NEXT J
5930    NEXT I
5940    STARTY%=STARTY%+DW%
5950    FINX%=FINX%-HW%
5960    K=AFTER%(K,1)
5970 NEXT L
5980 RETURN
5990 REM TAIL LL
6000 STARTY%=(LENGTH+1)*DW%+WIDTH1R%
6010 K=AFTER%(OM%,LL)
6020 FOR L=1 TO LLL-1
6030    FOR I=STARTY% TO STARTY%+DW%-1
```

```
6040       FOR J=WIDTH1% TO W12%-1
6050          PLAN%(I,J)=K
6060       NEXT J
6070    NEXT I
6080    X=AFTER%(K,1)
6090    STARTY%=STARTY%+DW%
6100 NEXT L
6110 REM FILL IN LAST ELEMENT OF TAIL LL
6120 FOR I=STARTY% TO W12R%-1
6130    FOR J=WIDTH1% TO W12%-1
6140       PLAN%(I,J)=K
6150    NEXT J
6160 NEXT I
6170 RETURN
```

# APPENDIX 2

2.1 Program BLOCK.

Written in Microsoft Basic Version 5.0.

Determines a maximal planar adjacency graph via constrained DELTAHEDRON or constrained SUPER_ DELTAHEDRON i.e. maximizes Relationship chart scores or minimizes transportation costs.

2.2 Program SPLAN.

Written in Microsoft Basic Version 5.0.

Constructs a block plan corresponding to the SUPER_DELTAHEDRON output from program BLOCK.

Note: this program has not been extensively tested; however, it has been successfully run on several randomly generated examples.

```
10 REM DELTAHEDRON AND SUPER_DELTAHEDRON HEURISTICS
20 DEFINT I-N
30 DIM BEN%(25,25),ORDER%(25),TRIANG%(46,3),SOLUTION%(25,25)
40 DIM EDGES%(305),R%(20),CARD%(10),DIST%(25),SAVED%(10), HMAXX%(3)
50 DIM CLASS%(10,3),LAYER%(10,25),H%(30),PAIR%(3,2),BETW%(3,10)
60 DIM VALID%(25),FUNDTRI%(10),ORIGIN%(10),FUND%(10,2),BADJ%(2)
70 DIM AND%(25),BEFORE%(25),AFTER%(25,10),KOUNT(10),NUM(25),STORE%(3,3)
80 DIM OTHERS%(25),DEG%(25),A%(25),DEGCON%(25),ROOTA%(25)
90 DIM SPATH%(25,25),HP%(3),HASH%(25),BENSUM%(25)
100 INPUT "NUMBER OF FACILITIES";N
110 LPRINT "NUMBER OF FACILITIES:"N
120 PRINT "DATA FOR THIS PROGRAM MAY BE IN THE FORM OF EITHER"
130 PRINT "  (1) A REL CHART,  OR"
140 PRINT "  (2) A MATRIX OF TRANSPORTATION COSTS"
150 PRINT "FACILITY AREAS ARE CONSIDERED EXPLICITLY ONLY IN CASE (2)"
160 PRINT
170 PRINT "  FOR TYPE (1), TYPE '1' "
180 PRINT "  FOR TYPE (2), TYPE '2' "
190 INPUT "TYPE OF DATA INPUT DESIRED": SUPER%
200 IF SUPER%=1 THEN 1180
210 PRINT "THIS PROGRAM ACCEPTS THREE FORMS OF INPUT"
220 PRINT
230 PRINT "  (1) DIRECT INPUT AT THE TERMINAL"
240 PRINT "  (2) READING A SET OF DATA STATEMENTS"
250 PRINT "  (3) GENERATION OF A RANDOM PROBLEM"
260 PRINT
270 PRINT "THE INPUT DATA IS IN THE FORM OF A TRANSPORTATION COST"
280 PRINT "MATRIX OF INTEGRAL VALUES, REPRESENTING THE TRAVEL COST"
290 PRINT "(PER UNIT TIME) BETWEEN FACILITIES I AND J (I(J), FOR"
300 PRINT "ALL FACILITY PAIRS. ALSO REQUIRED IS THE DESIRED AREA"
310 PRINT "FOR EACH FACILITY (2..N)"
320 PRINT
330 PRINT " FOR (1) TYPE 'I' "
340 PRINT " FOR (2) TYPE 'R' "
350 PRINT " FOR (3) TYPE 'G' "
360 PRINT
370 INPUT "TYPE OF DATA INPUT REQUIRED";P$
380 IF P$ = "I" GOTO 550
390 IF P$ = "R" GOTO 430
400 IF P$ = "G" GOTO 700
410 PRINT " INVALID OPTION. TRY AGAIN"
420 GOTO 370
430 FOR I=1 TO N
440    FOR J=I+1 TO N
450       READ BEN%(I,J)
460    NEXT J
470 NEXT I
480 REM READ FACILITY AREAS, EXCLUDING THE EXTERIOR FACILITY
490 FOR I=2.TO N
500    READ A%(I)
510    ROOTA%(I)=SQR(A%(I))
520    DEGCON%(I)=(ROOTA%(I)\2)+3
530 NEXT I
540 GOTO 970
550 FOR I=1 TO N
```

```
560    FOR J=I+1 TO N
570       PRINT "TRANSPORTATION COST BETWEEN FACILITIES" I " AND " J " :"
580       INPUT BEN%(I,J)
590       INPUT "IS THIS VALUE CORRECT (Y/N)" : P$
600       IF P$ = "N" GOTO 570
610    NEXT J
620 NEXT I
630 FOR I=2 TO N
640    PRINT "AREA OF FACILITY" I;
650    INPUT A%(I)
660    ROOTA%(I)=SQR(A%(I))
670    DEGCON%(I)=(ROOTA%(I)\2)+3
680 NEXT I
690 GOTO 970
700 RANDOMIZE
710 LPRINT "GENERATING TRANSPORTATION COST MATRIX RANDOMLY"
720 LPRINT
730 PRINT "THIS ROUTINE PROVIDES A NORMALLY DISTRIBUTED TRANSPORTATION"
740 PRINT "COST MATRIX WITH GIVEN MEAN AND VARIANCE. REQUIRED INPUT IS"
750 PRINT "THE MEAN, MU, AND THE STANDARD DEVIATION, SIGMA, OF THE"
760 PRINT "DESIRED DISTRIBUTION"
770 PRINT
780 INPUT "VALUE OF MU"; MU
790 LPRINT "VALUE OF MU:" MU
800 INPUT "VALUE OF SIGMA"; SIGMA%
810 LPRINT "VALUE OF SIGMA:" SIGMA%
820 FOR I=1 TO N
830    FOR J=I+1 TO N
840       BEN%(I,J)=INT(SQR(-2*LOG(RND))*COS(6.29319*RND)*SIGMA%+MU)
850       IF BEN%(I,J) < 0 THEN BEN%(I,J)=0
860    NEXT J
870 NEXT I
880 FOR J=2 TO N
890    I=INT(RND*10)
900    IF I <= 2 THEN ROOTA%(J)=6 : DEGCON%(J)=6 : GOTO 950
910    IF I <= 4 THEN ROOTA%(J)=8 : DEGCON%(J)=7 : GOTO 950
920    IF I <= 6 THEN ROOTA%(J)=10 : DEGCON%(J)=8 : GOTO 950
930    IF I <= 8 THEN ROOTA%(J)=12 : DEGCON%(J)=9 : GOTO 950
940    ROOTA%(J)=14 : DEGCON%(J)=10
950    A%(J)=ROOTA%(J)^2
960 NEXT J
970 LPRINT "TRANSPORTATION COST MATRIX:"
980 FOR I=1 TO N
990    FOR J=I+1 TO N
1000      BEN%(J,I)=BEN%(I,J)
1010   NEXT J
1020 NEXT I
1030 LPRINT
1040 FOR I=1 TO N
1050    FOR J=1 TO N
1060       LPRINT BEN%(I,J);
1070    NEXT J
1080    LPRINT
1090 NEXT I
1100 LPRINT
```

```
1110 LPRINT "FACILITY AREAS (2-N) :"
1120 LPRINT
1130 FOR I=2 TO N
1140    LPRINT A%(I):
1150 NEXT I
1160 LPRINT
1170 GOTO 2300
1180 PRINT "THIS PROGRAM ACCEPTS THREE FORMS OF INPUT"
1190 PRINT
1200 PRINT "   (1) DIRECT INPUT AT THE TERMINAL"
1210 PRINT "   (2) READING  A SET OF DATA STATEMENTS"
1220 PRINT "   (3) GENERATION OF A RANDOM PROBLEM"
1230 PRINT
1240 PRINT "THE INPUT FOR (1) AND (2) MUST BE IN THE FORM OF A"
1250 PRINT "RELATIONSHIP MATRIX, USING THE STANDARD CHARACTERS"
1260 PRINT "A,E,I,O,U,X, WITH THE CORRESPONDING NUMERICAL VALUES"
1270 PRINT "64,16,4,1,0,-128"
1280 PRINT
1290 PRINT " FOR (1) TYPE 'I' "
1300 PRINT " FOR (2) TYPE 'R' "
1310 PRINT " FOR (3) TYPE 'G' "
1320 PRINT
1330 INPUT "TYPE OF DATA INPUT REQUIRED"; P$
1340 IF P$="I" GOTO 1620
1350 IF P$="R" GOTO 1390
1360 IF P$="G" GOTO 2020
1370 PRINT " INVALID OPTION. TRY AGAIN"
1380 GOTO 1330
1390 FLAG%=0
1400 LPRINT
1410 LPRINT "RELATIONSHIP MATRIX IN THE FORM (I,J), FOR J ) I"
1420 LPRINT
1430 K=7
1440 FOR I=1 TO N
1450    LPRINT I " : ";
1460    LPRINT TAB(K)
1470    FOR J=I+1 TO N
1480       READ P$
1490       LPRINT P$ " ";
1500       IF P$="U" THEN BEN%(I,J)=0   : GOTO 1560
1510       IF P$="O" THEN BEN%(I,J)=1   : GOTO 1560
1520       IF P$="I" THEN BEN%(I,J)=4   : GOTO 1560
1530       IF P$="E" THEN BEN%(I,J)=16  : GOTO 1560
1540       IF P$="A" THEN BEN%(I,J)=64  : GOTO 1560
1550       IF P$="X" THEN BEN%(I,J)=-128 : FLAG%=1
1560    NEXT J
1570    K=K+2
1580    LPRINT
1590 NEXT I
1600 LPRINT
1610 GOTO 1870
1620 PRINT
1630 LPRINT
1640 LPRINT "RELATIONSHIP MATRIX IN THE FORM (I,J), FOR J ) I "
1650 LPRINT
```

```
1660 FLAG%=0
1670 K=7
1680 FOR I=1 TO N
1690    LPRINT I " : ";
1700    LPRINT TAB(K)
1710    FOR J=I+1 TO N
1720       PRINT "REL CHART SCORE FOR FACILITIES " I " , " J " :";
1730       INPUT P$
1740       LPRINT P$ " ";
1750       IF P$="U" THEN BEN%(I,J)=0  : GOTO 1810
1760       IF P$="O" THEN BEN%(I,J)=1  : GOTO 1810
1770       IF P$="I" THEN BEN%(I,J)=4  : GOTO 1810
1780       IF P$="E" THEN BEN%(I,J)=16 : GOTO 1810
1790       IF P$="A" THEN BEN%(I,J)=64 : GOTO 1810
1800       IF P$="X" THEN BEN%(I,J)=-128 : FLAG%=1
1810    NEXT J
1820    PRINT
1830    K=K+2
1840    LPRINT
1850 NEXT I
1860 LPRINT
1870 IF FLAG%=0 GOTO 1960
1880 FOR I=1 TO N
1890    FOR J=I+1 TO N
1900       BEN%(I,J)=BEN%(I,J)+128
1910    NEXT J
1920 NEXT I
1930 FOR I=1 TO N
1940    BEN%(I,I)=0
1950 NEXT I
1960 FOR I=1 TO N
1970    FOR J=I+1 TO N
1980       BEN%(J,I)=BEN%(I,J)
1990    NEXT J
2000 NEXT I
2010 GOTO 2210
2020 RANDOMIZE
2030 LPRINT "GENERATING RELATIONSHIP MATRIX RANDOMLY"
2040 PRINT
2050 PRINT "THIS ROUTINE WILL PROVIDE A RELATIONSHIP MATRIX (NORMALLY"
2060 PRINT "DISTRIBUTED) FOR TESTING PURPOSES, USING THE METHOD OF BOX AND"
2070 PRINT "MULLER. REQUIRED INPUT IS THE MEAN, MU, AND THE STANDARD"
2080 PRINT "DEVIATION, SIGMA, OF THE DESIRED DISTRIBUTION"
2090 PRINT
2100 INPUT "VALUE OF MU";MU
2110 LPRINT "VALUE OF MU:" MU
2120 INPUT "VALUE OF SIGMA";SIGMA%
2130 LPRINT "VALUE OF SIGMA:"SIGMA%
2140 FOR I=1 TO N
2150    FOR J=I+1 TO N
2160       BEN%(I,J)=INT(SQR(-2*LOG(RND))*COS(6.28319*RND)*SIGMA%+MU)
2170       IF BEN%(I,J) < 0 THEN BEN%(I,J)=0
2180       BEN%(J,I)=BEN%(I,J)
2190    NEXT J
2200 NEXT I
```

```
2210 LPRINT "RELATIONSHIP MATRIX IN NUMERICAL TERMS"
2220 LPRINT
2230 FOR I=1 TO N
2240    FOR J=1 TO N
2250       LPRINT BEN%(I,J):
2260    NEXT J
2270    LPRINT
2280 NEXT I
2290 LPRINT
2300 FOR I=1 TO N
2310    VALID%(I)=1
2320 NEXT I
2330 ORDER%(1)=1
2340 REM FIND THE BEST TETRAHEDRON INCLUDING THE EXTERIOR FACILITY
2350 MAX=-1
2360 FOR I=2 TO N-2
2370    FOR J=I+1 TO N-1
2380       FOR K=J+1 TO N
2390          SUM%=BEN%(I,1)+BEN%(J,1)+BEN%(K,1)+BEN%(I,J)+
                   BEN%(I,K)+BEN%(J,K)
2400          IF SUM% ) MAX THEN MAX=SUM%:
                 ORDER%(2)=I:ORDER%(3)=J:ORDER%(4)=K
2410       NEXT K
2420    NEXT J
2430 NEXT I
2440 TOTBEN%=MAX
2450 FOR I=2 TO 4
2460    VALID%(ORDER%(I))=0
2470 NEXT I
2480 LPRINT
2490 LPRINT "BEST TETRAHEDRON:"
2500 LPRINT
2510 FOR I=1 TO 4
2520    LPRINT ORDER%(I);
2530 NEXT I
2540 LPRINT
2550 FOR I=1 TO 4
2560    X%=ORDER%(I)
2570    FOR J=1 TO 4
2580       Y%=ORDER%(J)
2590       IF J(I THEN TRIANG%(I,J)=Y%
           ELSE IF J)I THEN TRIANG%(I,J-1)=Y%:
           SOLUTION%(X%,Y%)=1 : SOLUTION%(Y%,X%)=1
2600    NEXT J
2610 NEXT I
2620 FOR I=1 TO N
2630    H%(I)=0
2640 NEXT I
2650 FOR I=2 TO 4
2660    CLASS%(1,I-1)=ORDER%(I)
2670    LAYER(1,I-1)=ORDER%(I)
2680    DIST%(ORDER%(I))=1
2690 REM DEFINE HASH FUNCTION FOR THE FUNDAMENTAL TRIANGLE
2700    H%(ORDER%(I))=I-2
2710 NEXT I
```

```
2720 PAIR%(1,1)=ORDER%(2)
2730 PAIR%(1,2)=ORDER%(3)
2740 PAIR%(2,1)=ORDER%(2)
2750 PAIR%(2,2)=ORDER%(4)
2760 PAIR%(3,1)=ORDER%(3)
2770 PAIR%(3,2)=ORDER%(4)
2780 DIST%(1)=0
2790 CARD%(1)=3
2800 MAXDIST=1
2810 FUNDTRI%(1)=1
2820 TRINO%=4
2830 VERTNO%=4
2840 IF SUPER%=1 THEN 4150
2850 ROOTA%(1)=0
2860 DEGCON%(1)=N\2
2870 KK=5
2880 FOR I=2 TO N
2890    IF VALID%(I)=0 GOTO 2970
2900    ORDER%(KK)=I
2910    SUM%=0
2920    FOR J=1 TO N
2930       IF I <> J THEN SUM%=SUM%+BEN%(I,J)
2940    NEXT J
2950    BENSUM%(KK)=SUM%
2960    KK=KK+1
2970 NEXT I
2980 REM SORT THE N-4 REMAINING VERTICES ACCORDING TO THEIR COLUMN SUMS
2990 REM TO ASCERTAIN INSERTION ORDER...BUBBLESORT ORDER (5..N)
3000 REM ACCORDING TO BENSUM VALUES
3010 FLIPS%=1
3020 WHILE FLIPS%=1
3030    FLIPS%=0
3040    FOR I=5 TO N-1
3050       IF BENSUM%(I) < BENSUM%(I+1) THEN
              SWAP ORDER%(I),ORDER%(I+1): SWAP BENSUM%(I),BENSUM%(I+1):
              FLIPS%=1
3060    NEXT I
3070 WEND
3080 REM INSERTION ORDER IS ORDER(5)..ORDER(N)
3090 LPRINT
3100 LPRINT "INSERTION ORDER :"
3110 LPRINT
3120 FOR I=5 TO N
3130    LPRINT ORDER%(I);
3140 NEXT I
3150 LPRINT
3160 LPRINT
3170 SUMBEN=0
3180 FOR I=1 TO 4
3190    FOR J=I+1 TO 4
3200       XY%=(ROOTA%(ORDER%(I))+ROOTA%(ORDER%(J)))/2
3210       SPATH%(I,J)=XY%
3220       SPATH%(J,I)=SPATH%(I,J)
3230       SUMBEN=SUMBEN + XY%*BEN%(ORDER%(I),ORDER%(J))
3240    NEXT J
```

```
3250 NEXT I
3260 FOR I=1 TO 4
3270    DEG%(ORDER%(I))=3
3280 NEXT I
3290 FOR J=1 TO 4
3300    HASH%(ORDER%(J))=J
3310 NEXT J
3320 FOR I=5 TO N
3330    BENMIN=100000000#
3340    X%=ORDER%(I)
3350    HASH%(X%)=I
3360    IM1=I-1
3370    SUM=0!
3380    K=1
3390    OK%=0
3400    WHILE K (= TRINO%
3410    IF NOT ((DEG%(TRIANG%(K,1)) ( DEGCON%(TRIANG%(K,1))-1)
           AND  (DEG%(TRIANG%(K,2)) ( DEGCON%(TRIANG%(K,2))-1)
           AND  (DEG%(TRIANG%(K,3)) ( DEGCON%(TRIANG%(K,3))-1)) GOTO 3700
3420       SUM=0
3430       ACCEPT%=0
3440       FOR JJ=1 TO IM1
3450          OTHERS%(JJ)=1
3460       NEXT JJ
3470       FOR JJ=1 TO 3
3480          OTHERS%(HASH%(TRIANG%(K,JJ)))=0
3490       NEXT JJ
3500       FOR JJ=1 TO 3
3510          HP%(JJ)=HASH%(TRIANG%(K,JJ))
3520       NEXT JJ
3530       FOR JJ=1 TO 3
3540          XY%=(ROOTA%(X%)+ROOTA%(TRIANG%(K,JJ)))/2
3550          SPATH%(I,HP%(JJ))=XY%
3560          SPATH%(HP%(JJ),I)=XY%
3570          SUM=SUM+XY%*BEN%(X%,TRIANG%(K,JJ))
3580       NEXT JJ
3590       FOR KK=1 TO IM1
3600       IF OTHERS%(KK)=0 GOTO 3670
3610          M1=SPATH%(I,HP%(1))+SPATH%(HP%(1),KK)
3620          M2=SPATH%(I,HP%(2))+SPATH%(HP%(2),KK)
3630          M3=SPATH%(I,HP%(3))+SPATH%(HP%(3),KK)
3640          IF M1 ( M2 THEN MIN=M1 ELSE MIN=M2
3650          IF MIN ) M3 THEN MIN=M3
3660          SUM=SUM+MIN*BEN%(X%,ORDER%(KK))
3670       NEXT KK
3680       IF SUM ( BENMIN THEN V%=X%:GOSUB 5990
3690       IF ACCEPT%=1 THEN
                BENMIN=SUM : MAXTRI=K : MAXX=X% : OK%=1
3700       K=K+1
3710    WEND
3720    IF ((SUM ) .5) AND (OK%=1)) THEN 3810
3730    FOR K=2 TO N
3740    IF DEG%(K) () DEGCON%(K)-1 GOTO 3780
3750       LPRINT "** PERTURBING DEGREE CONSTRAINT ON VERTEX" K " **"
3760       LPRINT
```

```
3770       DEGCON%(K)=DEGCON%(K)+1
3780    NEXT K
3790    BENMIN=10000000#
3800    GOTO 3370
3810    FOR JJ=1 TO IM1
3820       OTHERS%(JJ)=1
3830    NEXT JJ
3840    FOR JJ=1 TO 3
3850       OTHERS%(HASH%(TRIANG%(MAXTRI,JJ)))=0
3860    NEXT JJ
3870    SUMBEN=SUMBEN+BENMIN
3880    FOR JJ=1 TO 3
3890       HP%(JJ)=HASH%(TRIANG%(MAXTRI,JJ))
3900    NEXT JJ
3910    FOR JJ=1 TO 3
3920       XY%=(ROOTA%(X%)+ROOTA%(TRIANG%(MAXTRI,JJ)))/2
3930       SPATH%(I,HP%(JJ))=XY%
3940       SPATH%(HP%(JJ),I)=XY%
3950    NEXT JJ
3960    FOR KK=1 TO IM1
3970    IF OTHERS%(KK)=0 THEN 4050
3980       M1=SPATH%(I,HP%(1))+SPATH%(HP%(1),KK)
3990       M2=SPATH%(I,HP%(2))+SPATH%(HP%(2),KK)
4000       M3=SPATH%(I,HP%(3))+SPATH%(HP%(3),KK)
4010       IF M1 ( M2 THEN MIN=M1 ELSE MIN=M2
4020       IF MIN ) M3 THEN MIN=M3
4030       SPATH%(I,KK)=MIN
4040       SPATH%(KK,I)=MIN
4050    NEXT KK
4060    DEG%(X%)=3
4070    FOR J=1 TO 3
4080       Y%=TRIANG%(MAXTRI,J)
4090       DEG%(Y%)=DEG%(Y%)+1
4100    NEXT J
4110    GOSUB 7080
4120 NEXT I
4130 GOSUB 7310
4140 GOTO 4930
4150 WHILE VERTNO% ( N
4160    MAX=-1
4170    FOR I=2 TO N
4180    IF VALID%(I)=0 THEN 4310
4190       K=1
4200       WHILE K (= TRINO%
4210          ACCEPT%=0
4220          SUM%=0
4230          FOR J=1 TO 3
4240             L=TRIANG%(K,J)
4250             SUM%=SUM%+BEN%(I,L)
4260          NEXT J
4270          IF SUM% ) MAX THEN VX=I : GOSUB 5990
4280          IF ACCEPT%=1 THEN
                 MAX=SUM% : MAXTRI=K : MAXX=I
4290          K=K+1
4300       WEND
```

```
4310    NEXT I
4320    GOSUB 7080
4330    VERTNO%=VERTNO%+1
4340    VALID%(MAXX)=0
4350    TOTBEN%=TOTBEN%+MAX
4360 WEND
4370 LPRINT
4380 IF FLAG%=0 GOTO 4450
4390 TOTBEN%=TOTBEN%-(3*N-6)*128
4400 FOR I=1 TO N
4410    FOR J=1 TO N
4420       BEN%(I,J)=BEN%(I,J)-128
4430    NEXT J
4440 NEXT I
4450 LPRINT "TOTAL DELTAHEDRON ADJACENCY SCORE IS" TOTBEN%
4460 LPRINT
4470 REM WRITE OUT SOLUTION (INCIDENCE) MATRIX
4480 LPRINT "INCIDENCE MATRIX:"
4490 LPRINT
4500 FOR I=1 TO N
4510    FOR J=1 TO N
4520       LPRINT SOLUTION%(I,J);
4530    NEXT J
4540    LPRINT
4550 NEXT I
4560 LPRINT
4570 REM SORT THE EDGES ACCORDING TO WEIGHT TO GET 3N-6 BOUND
4580 M=0
4590 FOR I=1 TO N
4600    FOR J=I+1 TO N
4610       M=M+1
4620       EDGES%(M)=BEN%(I,J)
4630    NEXT J
4640 NEXT I
4650 R%(1)=M\2
4660 T%=INT(LOG(M)/LOG(2))+1
4670 FOR I=1 TO T%-1
4680    R%(I+1)=INT(.75*R%(I))
4690 NEXT I
4700 FOR L=1 TO T%
4710    INC=R%(L)
4720    FOR I=1 TO M-INC
4730       IF EDGES%(I) ( EDGES%(I+INC) THEN SWAP EDGES%(I),EDGES%(I+INC)
4740    NEXT I
4750 NEXT L
4760 L=M-1
4770 WHILE L ) 0
4780    K=0
4790    FOR I=1 TO L
4800       IF EDGES%(I) ( EDGES%(I+1) THEN SWAP EDGES%(I),EDGES%(I+1):K=I
4810    NEXT I
4820    L=K-1
4830 WEND
4840 LPRINT
4850 BOUND%=0
```

```
4860 FOR I=1 TO 3*N-6
4870    BOUND%=BOUND%+EDGES%(I)
4880 NEXT I
4890 LPRINT
4900 LPRINT "BEST POSSIBLE SOLUTION BOUND IS" BOUND%
4910 LPRINT
4920 LPRINT "DELTAHEDRON SOLUTION RATIO IS " TOTBEN%/BOUND%
4930 LPRINT
4940 LPRINT "DISTANCE CLASSES:"
4950 LPRINT
4960 FOR I=1 TO MAXDIST
4970    FOR J=1 TO CARD%(I)
4980       LPRINT LAYER(I,J);
4990    NEXT J
5000    LPRINT
5010 NEXT I
5020 LPRINT
5030 REM THE FOLLOWING STATEMENTS TRANSFER ALL RELEVANT DATA REQUIRED
5040 REM FOR THE LAYOUT PHASE TO DISK IN PREPARATION FOR READING BY
5050 REM EITHER OF THE SECOND PROGRAMS (PLAN AND SPLAN). IF SUPER_
5060 REM DELTAHEDRON IS REQUIRED, INFO IS READ INTO FILE "PLANDATA".
5070 REM IF DELTAHEDRON IS REQUIRED, INFO IS READ INTO FILE "PROBDATA".
5080 REM THUS THE PROGRAM MAY BE RUN ON A 64K MICROCOMPUTER
5090 IF SUPER%=1 THEN 5120
5100 OPEN "O",#1,"PLANDAT2"
5110 GOTO 5130
5120 OPEN "O",#1,"PROBDATA"
5130 PRINT #1,N
5140 PRINT #1,MAXDIST
5150 FOR I=1 TO N
5160    FOR J=1 TO N
5170       PRINT #1,SOLUTION%(I,J)
5180    NEXT J
5190 NEXT I
5200 FOR I=1 TO MAXDIST
5210    PRINT #1,CARD%(I)
5220 NEXT I
5230 FOR I=1 TO MAXDIST
5240    FOR J=1 TO 3
5250       PRINT #1,CLASS%(I,J)
5260    NEXT J
5270 NEXT I
5280 FOR I=1 TO N
5290    PRINT #1,BEFORE%(I)
5300 NEXT I
5310 FOR I=1 TO N
5320    PRINT #1,ANO%(I)
5330 NEXT I
5340 FOR I=1 TO MAXDIST
5350    PRINT #1,KOUNT(I)
5360 NEXT I
5370 FOR I=1 TO N
5380    FOR J=1 TO ANO%(I)
5390       PRINT #1,AFTER%(I,J)
5400    NEXT J
```

```
5410 NEXT I
5420 FOR I=1 TO MAXDIST
5430    PRINT #1,FUNDTRI%(I)
5440 NEXT I
5450 FOR I=1 TO MAXDIST
5460    PRINT #1,ORIGIN%(I)
5470 NEXT I
5480 FOR I=1 TO 3
5490    PRINT #1,NUM(I)
5500 NEXT I
5510 FOR I=1 TO 3
5520    FOR J=1 TO NUM(I)
5530      PRINT #1,BETW%(I,J)
5540    NEXT J
5550 NEXT I
5560 FOR I=1 TO N
5570    PRINT #1,H%(I)
5580 NEXT I
5590 FOR I=1 TO KOUNT(MAXDIST)
5600    PRINT #1,SAVED%(I)
5610 NEXT I
5620 IF SUPER%=1 THEN 5720
5630 FOR I=2 TO N
5640    PRINT #1,A%(I)
5650 NEXT I
5660 FOR I=1 TO CARD%(MAXDIST)
5670    PRINT #1,LAYER(MAXDIST,I)
5680 NEXT I
5690 CLOSE #1
5700 RUN "SPLAN"
5710 END
5720 CLOSE #1
5730 RUN "PLAN"
5740 END
5750 D%=DIST%(MAXX)
5760 IF BEFORE%(A%)=B% THEN SWAP A%,B%
5770 REM DETERMINE THE INDICES OF THE TWO VERTICES OF CLASS(D-1,*)
5780 REM TO WHICH B IS ADJACENT
5790 KK=1
5800 FOR K=1 TO 3
5810    IF SOLUTION%(B%,CLASS%(D%-1,K))=1 THEN
           BADJ%(KK)=K : KK=KK+1
5820 NEXT K
5830 REM SWAP INDICES IF NECESSARY
5840 IF ((BADJ%(1)=1) AND (BADJ%(2)=3)) THEN SWAP BADJ%(1),BADJ%(2)
5850 REM CHECK WHICH VERTEX MAXX IS ADJACENT TO
5860 REM IF THE FIRST, ORIENTATION IS (A,MAXX,B)
5870 REM IF THE SECOND, ORIENTATION IS (A,B,MAXX)
5880 IF SOLUTION%(MAXX,CLASS%(D%-1,BADJ%(1)))=1 THEN
        CLASS%(D%,1)=A%:CLASS%(D%,2)=MAXX:CLASS%(D%,3)=B%
     ELSE CLASS%(D%,1)=A%:CLASS%(D%,2)=B%:CLASS%(D%,3)=MAXX
5890 LPRINT
5900 LPRINT "ORIENTED CYCLE:"
5910 LPRINT
5920 LPRINT D% ":";
```

```
5930 FOR J=1 TO 3
5940    LPRINT CLASS%(D%,J);
5950 NEXT J
5960 LPRINT
5970 LPRINT
5980 RETURN
5990 REM PREVENT CREATION OF A SECOND TRIANGLE IN THE DISTANCE CLASS
6000 A%=TRIANG%(K,1)
6010 B%=TRIANG%(K,2)
6020 C%=TRIANG%(K,3)
6030 IF DIST%(A%) () DIST%(B%) THEN
        IF DIST%(B%)= DIST%(C%) THEN SWAP A%,C%
        ELSE SWAP B%,C%
6040 REM STOP CREATION OF TRIANGLES IN DISTANCE CLASS ONE WHICH ARE
6050 REM NOT OF AN APPROPRIATE TYPE ("CLASS 1")
6060 IF ((H%(A%)=-1) OR (H%(B%)=-1) OR (H%(C%)=-1)) THEN 6310
6070 REM PREVENT INSERTION IN ANY "CLASS 1" TRIANGLE OF DISTANCE CLASS
6080 REM ONE...THE ONLY ALLOWABLE SUCH TRIANGLE IS THE FUNDAMENTAL
6090 REM TRIANGLE, CONSTRUCTED AS TRIANG(1,*)
6100 IF DIST%(A%)=1 THEN
       IF ((DIST%(A%)=DIST%(B%)) AND (DIST%(B%)=DIST%(C%))) THEN
          IF K () 1 THEN 6310 ELSE 6300
6110 REM PREVENT CREATION OF TRIANGLES IN OTHER DISTANCE CLASSES UNLESS
6120 REM THE FUNDAMENTAL TRIANGLE OF THAT CLASS IS YET TO BE DEFINED
6130 IF DIST%(A%) ) 1 THEN
       IF ((DIST%(A%)=DIST%(C%)+1) AND (FUNDTRI%(DIST%(A%))=1)) GOTO 6310
6140 REM CONSTRAIN TO TWO DISTANCE CLASSES ONLY
6150 IF MAXDIST=2 THEN
       IF ((DIST%(A%)=DIST%(B%)) AND (DIST%(B%)=DIST%(C%))) GOTO 6310
6160 REM DETERMINE DISTANCE OF V TO EXTERIOR FACILITY IMPLIED BY
6170 REM ITS INSERTION IN TRIANGLE K
6180 MINIM=N
6190 FOR L=1 TO 3
6200    D%=DIST%(TRIANG%(K,L))
6210    IF D% ( MINIM THEN MINIM=D%
6220 NEXT L
6230 DIST%(V%)=MINIM+1
6240 REM NOW DETERMINE WHETHER ADDING V TO DISTANCE CLASS DIST(I)
6250 REM IS VALID IN TERMS OF THE CONSTRAINED APPROACH...I.E. DOES
6260 REM THE DISTANCE CLASS REMAIN CONNECTED?
6270 IF CARD%(DIST%(V%))=0 THEN 6300
6280 IF ((DIST%(A%)=DIST%(B%)) AND (DIST%(B%)=DIST%(C%))) THEN
       IF CARD%(DIST%(V%)) ) 0 THEN 6310
6290 REM ACCEPT TRIANG(K) AS TRIANGLE FOR INSERTION
6300 ACCEPT%=1
6310 RETURN
6320 REM ELEMENTS OF INSERTION TRIANGLE ARE TRIANG(MAXTRI,*)
6330 A%=TRIANG%(MAXTRI,1)
6340 B%=TRIANG%(MAXTRI,2)
6350 C%=TRIANG%(MAXTRI,3)
6360 IF DIST%(A%) () DIST%(B%) THEN
       IF DIST%(B%)=DIST%(C%) THEN SWAP A%,C%
         ELSE SWAP B%,C%
6370 REM NOW HAVE ONE OF THE FOLLOWING CASES
6380 REM     (1) DIST(A)=DIST(B)=DIST(C)+1
```

```
6390 REM     (2) DIST(A)=DIST(B)=DIST(C)-1
6400 REM     (3) DIST(A)=DIST(B)=DIST(C)
6410 DA%=DIST%(A%)
6420 DB%=DIST%(B%)
6430 DC%=DIST%(C%)
6440 IF DA%=DC%+1 THEN 6510
6450 IF DA%=DC%-1 THEN 7020
6460 REM OTHERWISE, WE HAVE CASE (3)
6470 DIST%(MAXX)=DA%+1
6480 MAXDIST=MAXDIST+1
6490 ORIGIN%(DIST%(MAXX))=MAXX
6500 GOTO 7070
6510 REM CASE (1)
6520 IF C%=1 GOTO 6710
6530 FUNDTRI%(DA%)=1
6540 DIST%(MAXX)=DA%
6550 REM DEFINE AND ORIENT CLASS(DA,*)
6560 GOSUB 5750
6570 IF BEFORE%(A%)=B% THEN Q%=B%:FUND%(DA%,1)=B%:
                         FUND%(DA%,2)=A%: GOTO 6610
6580 Q%=A%
6590 FUND%(DA%,1)=A%
6600 FUND%(DA%,2)=B%
6610 IF BEFORE%(Q%)=0 GOTO 7070
6620 SAVEQ%=Q%
6630 WHILE Q% <> ORIGIN%(DA%)
6640     AFTER%(Q%,1)=BEFORE%(Q%)
6650     SAVEQ%=Q%
6660     KOUNT(DA%)=KOUNT(DA%)+1
6670     SAVED%(KOUNT(DA%))=BEFORE%(Q%)
6680     Q%=BEFORE%(Q%)
6690 WEND
6700 GOTO 7070
6710 IF H%(A%) <> -2 THEN 6750
6720 FOR KK=1 TO 3
6730    IF STORE%(KK,1)=A% THEN L=H%(STORE%(KK,2))+H%(STORE%(KK,3)):GOTO 6890
6740 NEXT KK
6750 IF H%(B%) <> -2 THEN 6790
6760 FOR KK=1 TO 3
6770    IF STORE%(KK,1)=B% THEN L=H%(STORE%(KK,2))+H%(STORE%(KK,3)):GOTO 6890
6780 NEXT KK
6790 L=H%(A%)+H%(B%)
6800 IF NUM(L) > 0 GOTO 6880
6810 STORE%(L,1)=MAXX
6820 STORE%(L,2)=A%
6830 STORE%(L,3)=B%
6840 H%(MAXX)=-2
6850 NUM(L)=1
6860 BETW%(L,1)=MAXX
6870 GOTO 7000
6880 IF NUM(L) > 1 GOTO 6950
6890 FOR LL=1 TO 2
6900    IF A%=PAIR%(L,LL) THEN
          HMAXX%(L)=H%(PAIR%(L,LL MOD 2 +1)) : GOTO 6950
6910 NEXT LL
```

```
6920 FOR LL=1 TO 2
6930    IF B%=PAIR%(L,LL) THEN
           HMAXX%(L)=H%(PAIR%(L,LL MOD 2 +1)) : GOTO 6950
6940 NEXT LL
6950 H%(MAXX)=HMAXX%(L)
6960 H%(STORE%(L,1))=-1
6970 STORE%(L,1)=MAXX
6980 NUM(L)=NUM(L)+1
6990 BETW%(L,NUM(L))=MAXX
7000 DIST%(MAXX)=1
7010 GOTO 7070
7020 REM CASE (2)
7030 ANO%(C%)=ANO%(C%)+1
7040 AFTER%(C%,ANO%(C%))=MAXX
7050 BEFORE%(MAXX)=C%
7060 DIST%(MAXX)=DC%
7070 RETURN
7080 FOR J=1 TO 3
7090    SOLUTION%(MAXX,TRIANG%(MAXTRI,J))=1
7100    SOLUTION%(TRIANG%(MAXTRI,J),MAXX)=1
7110 NEXT J
7120 GOSUB 6320
7130 TRINO%=TRINO%+1
7140 LPRINT "INSERTING VERTEX" MAXX "IN TRIANGLE ";
7150 FOR L=1 TO 3
7160    LPRINT TRIANG%(MAXTRI,L);
7170 NEXT L
7180 LPRINT
7190 LPRINT
7200 TRIANG%(TRINO%,1)=TRIANG%(MAXTRI,1)
7210 TRIANG%(TRINO%,2)=TRIANG%(MAXTRI,2)
7220 TRIANG%(TRINO%,3)=MAXX
7230 TRINO%=TRINO%+1
7240 TRIANG%(TRINO%,1)=TRIANG%(MAXTRI,1)
7250 TRIANG%(TRINO%,2)=TRIANG%(MAXTRI,3)
7260 TRIANG%(TRINO%,3)=MAXX
7270 TRIANG%(MAXTRI,1)=MAXX
7280 CARD%(DIST%(MAXX))=CARD%(DIST%(MAXX))+1
7290 LAYER(DIST%(MAXX),CARD%(DIST%(MAXX)))=MAXX
7300 RETURN
7310 LPRINT "TRIANGLE LIST:"
7320 LPRINT
7330 FOR I=1 TO TRINO%
7340    FOR J=1 TO 3
7350       LPRINT TRIANG%(I,J);
7360    NEXT J
7370    LPRINT
7380 NEXT I
7390 LPRINT
7400 LPRINT "INCIDENCE MATRIX:"
7410 LPRINT
7420 FOR I=1 TO N
7430    FOR J=1 TO N
7440       LPRINT SOLUTION%(I,J);
7450    NEXT J
```

```
7460    LPRINT
7470 NEXT I
7480 LPRINT
7490 LPRINT "SUPER_DELTAHEDRON SOLUTION SCORE : " SUMBEN
7500 LPRINT
7510 LPRINT "DEGREE CONSTRAINTS (2..N) :"
7520 LPRINT
7530 FOR I=2 TO N
7540    LPRINT DEGCON%(I);
7550 NEXT I
7560 LPRINT
7570 LPRINT
7580 LPRINT "DEGREES ACHIEVED (2..N) :"
7590 LPRINT
7600 FOR I=2 TO N
7610    LPRINT DEG%(I);
7620 NEXT I
7630 LPRINT
7640 LPRINT
7650 LPRINT "SHORTEST PATH MATRIX:"
7660 LPRINT
7670 FOR I=1 TO N
7680    FOR J=1 TO N
7690       LPRINT SPATH%(HASH%(I),HASH%(J));
7700    NEXT J
7710    LPRINT
7720 NEXT I
7730 RETURN
```

```
10 REM OUTPUT BLOCK PLAN USING INPUT FROM "BLOCK"
20 DEFINT I-N
30 DIM SOLUTION%(30,30),CHAR$(30),A%(30),LAYER(2,30)
40 DIM CARD%(10),TAILLENG%(3),PLAN%(60,60)
50 DIM CLASS%(10,3),BETW%(3,10),H%(30)
60 DIM FUNDTRI%(10),ORIGIN%(10),SAVED%(10)
70 DIM ANO%(30),BEFORE%(30),AFTER%(30,10),KOUNT(10),NUM(30)
80 OPEN "I",#1,"PLANDAT2"
90 INPUT #1,N
100 INPUT #1,MAXDIST
110 FOR I=1 TO N
120    FOR J=1 TO N
130       INPUT #1,SOLUTION%(I,J)
140    NEXT J
150 NEXT I
160 FOR I=1 TO MAXDIST
170    INPUT #1,CARD%(I)
180 NEXT I
190 FOR I=1 TO MAXDIST
200    FOR J=1 TO 3
210       INPUT #1,CLASS%(I,J)
220    NEXT J
230 NEXT I
240 FOR I=1 TO N
250    INPUT #1,BEFORE%(I)
260 NEXT I
270 FOR I=1 TO N
280    INPUT #1,ANO%(I)
290 NEXT I
300 FOR I=1 TO MAXDIST
310    INPUT #1,KOUNT(I)
320 NEXT I
330 FOR I=1 TO N
340    FOR J=1 TO ANO%(I)
350       INPUT #1,AFTER%(I,J)
360    NEXT J
370 NEXT I
380 FOR I=1 TO MAXDIST
390    INPUT #1,FUNDTRI%(I)
400 NEXT I
410 FOR I=1 TO MAXDIST
420    INPUT #1,ORIGIN%(I)
430 NEXT I
440 FOR I=1 TO 3
450    INPUT #1,NUM(I)
460 NEXT I
470 FOR I=1 TO 3
480    FOR J=1 TO NUM(I)
490       INPUT #1,BETW%(I,J)
500    NEXT J
510 NEXT I
520 FOR I=1 TO N
530    INPUT #1,H%(I)
540 NEXT I
550 FOR I=1 TO KOUNT(MAXDIST)
```

```
560    INPUT #1,SAVED%(I)
570 NEXT I
580 FOR I=2 TO N
590    INPUT #1,A%(I)
600 NEXT I
610 FOR I=1 TO CARD%(MAXDIST)
620    INPUT #1,LAYER(MAXDIST,I)
630 NEXT I
640 CLOSE #1
650 FOR I=1 TO N
660    READ CHAR$(I)
670 NEXT I
680 DATA H,I,(,O,V,X,\,$,*,+,-,/,&,=,@,),#,%,A,B,C,D,E,F,G,J,K,L,M,N,P
690 INPUT "RATIO OF BUILDING LENGTH TO WIDTH ()=1)"; RATIO
700 INPUT "HORIZONTAL SIZE OF PLAN MATRIX DESIRED"; WID%
710 REM ARTIFICIALLY CREATE SECOND DISTANCE CLASS WITH ZERO ELEMENTS
720 REM FOR THE CASE SUCH THAT CARD(2)=0
730 IF CARD%(2)=0 THEN MAXDIST=2
740 IF CARD%(MAXDIST) ) 1 GOTO 790
750 TOP%=CLASS%(MAXDIST-1,1)
760 LEFT=CLASS%(MAXDIST-1,2)
770 RIGHT%=CLASS%(MAXDIST-1,3)
780 GOTO 1330
790 IF FUNDTRI%(MAXDIST)=1 GOTO 1060
800 IF ANO%(ORIGIN%(MAXDIST)) ) 1 GOTO 920
810 K=AFTER%(ORIGIN%(MAXDIST),1)
820 FOR L=1 TO 3
830    IF SOLUTION%(K,CLASS%(MAXDIST-1,L))()1 GOTO 850
840 NEXT L
850 TOP%=CLASS%(MAXDIST-1,L)
860 IF L=3 THEN L=1 ELSE L=L+1
870 LEFT=CLASS%(MAXDIST-1,L)
880 L=L+1
890 IF L=4 THEN L=1
900 RIGHT%=CLASS%(MAXDIST-1,L)
910 GOTO 1330
920 IF ANO%(ORIGIN%(MAXDIST)) ) 2 GOTO 750
930 K=AFTER%(ORIGIN%(MAXDIST),1)
940 L=AFTER%(ORIGIN%(MAXDIST),2)
950 FOR I=1 TO 3
960    IF ((SOLUTION%(K,CLASS%(MAXDIST-1,I))=1) AND
            (SOLUTION%(L,CLASS%(MAXDIST-1,I)))=1) GOTO 980
970 NEXT I
980 IF I=3 THEN I=1 ELSE I=I+1
990 TOP%=CLASS%(MAXDIST-1,I)
1000 IF I=3 THEN I=1 ELSE I=I+1
1010 LEFT=CLASS%(MAXDIST-1,I)
1020 I=I+1
1030 IF I=4 THEN I=1
1040 RIGHT%=CLASS%(MAXDIST-1,I)
1050 GOTO 1330
1060 REM FIRSTLY DETERMINE INNER SANCTUM POSITIONS
1070 FOR K=1 TO 3
1080 IF ((BEFORE%(CLASS%(MAXDIST,K))=0) AND
        (CLASS%(MAXDIST,K) () ORIGIN%(MAXDIST))) GOTO 1100
```

```
1090 NEXT K
1100 INRIGHT=CLASS%(MAXDIST,K)
1110 L=K MOD 3 +1
1120 LL=L MOD 3 +1
1130 IF BEFORE%(CLASS%(MAXDIST,L))=CLASS%(MAXDIST,LL)
       THEN C%=L ELSE C%=LL
1140 INLEFT=CLASS%(MAXDIST,C%)
1150 FOR I=1 TO 3
1160    IF ((I () K) AND (I () C%)) THEN KL=I
1170 NEXT I
1180 INTOP=CLASS%(MAXDIST,KL)
1190 REM NOW DEFINE CLASS(1) POSITIONS RELATIVE TO THE INNER SANCTUM
1200 FOR I=1 TO 3
1210    IF SOLUTION%(INRIGHT,CLASS%(MAXDIST-1,I))=1
          THEN L=I : GOTO 1230
1220 NEXT I
1230 RIGHT%=CLASS%(MAXDIST-1,L)
1240 FOR I=1 TO 3
1250    IF I=L GOTO 1270
1260    IF SOLUTION%(CLASS%(MAXDIST-1,I),INLEFT)=1 THEN LL=I:GOTO 1280
1270 NEXT I
1280 LEFT=CLASS%(MAXDIST-1,LL)
1290 FOR I=1 TO 3
1300    IF ((I () L) AND (I () LL)) THEN KL=I
1310 NEXT I
1320 TOP%=CLASS%(MAXDIST-1,KL)
1330 REM THE AREA OCCUPIED BY EACH DISTANCE CLASS MUST BE CONCOMITANT
1340 REM WITH THE SUM OF THE AREAS OF THE ELEMENTS OF EACH CLASS
1350 REM FIRSTLY IT IS NECESSARY TO SCALE THESE AREAS IN RELATION TO
1360 REM THE SIZE OF THE PLAN MATRIX DESIRED
1370 SUM%=0
1380 FOR I=1 TO N
1390    SUM%=SUM%+A%(I)
1400 NEXT I
1410 FOR I=1 TO N
1420    A%(I)=A%(I)*3600*RATIO/SUM%
1430 NEXT I
1440 REM SUM THE AREAS OF THE COMPONENT PARTS OF THE PLAN
1450 P3Y%=INT(WID%*RATIO)
1460 P1Y%=A%(TOP%)\WID%+1
1470 SUMTL%=A%(LEFT)
1480 HTL%=H%(TOP%)+H%(LEFT)
1490 FOR K=1 TO NUM(HTL%)
1500    SUMTL%=SUMTL%+A%(BETW%(HTL%,K))
1510 NEXT K
1520 P1X%=SUMTL%\(P3Y%-P1Y%)
1530 SUMI%=0
1540 FOR K=1 TO CARD%(MAXDIST)
1550    SUMI%=SUMI%+A%(LAYER(MAXDIST,K))
1560 NEXT K
1570 SUMTR%=0
1580 SUMLR%=0
1590 HTR%=H%(TOP%)+H%(RIGHT%)
1600 FOR K=1 TO NUM(HTR%)
1610    SUMTR%=SUMTR%+A%(BETW%(HTR%,K))
```

```
1620 NEXT K
1630 HLR%=H%(LEFT)+H%(RIGHT%)
1640 FOR K=1 TO NUM(HLR%)
1650    SUMLR%=SUMLR%+A%(BETW%(HLR%,K))
1660 NEXT K
1670 REM SET UP THE DIMENSIONS OF THE INNER SANCTUM, UNDER THE ASSUMPTION
1680 REM THAT THE DIMENSIONS OF THE COMPONENT PARTS ARE PROPORTIONAL
1690 REM TO THEIR AREAS
1700 REM SPECIAL CASE FOR WHEN CARD(MAXDIST)=0
1710 IF CARD%(MAXDIST) <> 0 THEN 1750
1720 P2X%=P1X%
1730 P2Y%=P1Y%
1740 GOTO 1890
1750 IF ((SUMLR% <> 0) OR (SUMTR% <> 0)) THEN 1780
1760 GAMMA%=SQR(SUMI%/(SUMI%+A%(RIGHT%)))*(P3Y%-P1Y%)
1770 GOTO 1870
1780 IF SUMLR% <> 0 THEN 1810
1790 GAMMA%=SQR(SUMI%/(SUMI%+SUMTR%+A%(RIGHT%)))*(P3Y%-P1Y%)
1800 GOTO 1870
1810 IF SUMTR% <> 0 THEN 1840
1820 GAMMA%=SQR(SUMI%/(SUMI%+SUMLR%+A%(RIGHT%)))*(P3Y%-P1Y%)
1830 GOTO 1870
1840 ALPHA=SUMLR%*(WID%-P1X%)/SUMTR%-P3Y%+P1Y%
1850 BETA=SUMI%*SUMLR%/SUMTR%
1860 GAMMA%=(-ALPHA+SQR(ALPHA^2+4*BETA))/2
1870 P2Y%=P1Y%+GAMMA%
1880 P2X%=SUMI%/GAMMA%+P1X%
1890 REM FILL IN TOP
1900 FOR I=1 TO P1Y%-1
1910    FOR J=1 TO WID%
1920       PLAN%(I,J)=TOP%
1930    NEXT J
1940 NEXT I
1950 REM FILL IN LEFT (INITIALLY, THE COMPLETE RECTANGLE)
1960 FOR I=P1Y% TO P3Y%
1970    FOR J=1 TO P1X%-1
1980       PLAN%(I,J)=LEFT
1990    NEXT J
2000 NEXT I
2010 REM FILL IN BETWEEN LEFT AND TOP
2020 IF NUM(HTL%)=0 GOTO 2590
2030 IF NUM(HTL%)=1 GOTO 2310
2040 IF SOLUTION%(BETW%(HTL%,2),TOP%)=1 GOTO 2310
2050 REM OTHERWISE TAIL HTL IS ADJACENT TO LEFT
2060 REM INITIALLY ALLOW ONLY 1 COLUMN WIDTH 'CORRIDOR'
2070 STARTY%=P1Y%
2080 FINX%=P1X%-2
2090 B%=0
2100 FOR K=1 TO NUM%(HTL%)
2110    KK=BETW%(HTL%,K)
2120    AKK%=A%(KK)
2130    FOR I=STARTY% TO P3Y%
2140       FOR J=1 TO FINX%
2150          PLAN%(I,J)=KK
2160          B%=B%+1
```

```
2170          IF B% )= AKK% THEN 2200
2180       NEXT J
2190    NEXT I
2200    IF J=FINX% THEN 2270
2210    IF K=NUM(HTL%) THEN KK=BETW%(HTL%,K) ELSE KK=BETW%(HTL%,K+1)
2220    B%=0
2230    FOR L=J+1 TO FINX%
2240       PLAN%(I,J)=KK
2250       B%=B%+1
2260    NEXT L
2270    IF K=NUM(HTL%) THEN 2300
2280    STARTY%=I+1
2290 NEXT K
2300 GOTO 2590
2310 REM TAIL HTL IS ADJACENT TO TOP
2320 SUMTL%=SUMTL%-A%(LEFT)
2330 FINX%=P1X%-2
2340 REM AGAIN IMPLIES 'CORRIDOR' EFFECT
2350 B%=0
2360 K=BETW%(HTL%,1)
2370 FOR I=P1Y% TO P3Y%
2380    FOR J=1 TO FINX%
2390       PLAN%(I,J)=K
2400       B%=B%+1
2410       IF B% )= SUMTL% GOTO 2440
2420    NEXT J
2430 NEXT I
2440 IF NUM(HTL%)=1 GOTO 2590
2450 SUMTL%=SUMTL%-A%(BETW%(HTL%,1))
2460 FOR K=2 TO NUM(HTL%)
2470    KK=BETW%(HTL%,K)
2480    FINX%=FINX%-1
2490    B%=0
2500    FOR I=P1Y% TO P3Y%
2510       FOR J=1 TO FINX%
2520          PLAN%(I,J)=KK
2530          B%=B%+1
2540          IF B% )= SUMTL% THEN 2570
2550       NEXT J
2560    NEXT I
2570    SUMTL%=SUMTL%-B%
2580 NEXT K
2590 REM FILL IN RIGHT
2600 FOR I=P1Y% TO P3Y%
2610    FOR J=P2X% TO WID%
2620       PLAN%(I,J)=RIGHT%
2630    NEXT J
2640 NEXT I
2650 FOR I=P2Y% TO P3Y%
2660    FOR J=P1X% TO P2X%-1
2670       PLAN%(I,J)=RIGHT%
2680    NEXT J
2690 NEXT I
2700 REM FILL IN BETWEEN TOP AND RIGHT
2710 IF NUM(HTR%)=0 GOTO 3240
```

```
2720 IF NUM(HTR%)=1 GOTO 2990
2730 IF SOLUTION%(BETW%(HTR%,2),TOP%)=1 GOTO 2980
2740 STARTX%=P2X%+1
2750 STARTY%=P1Y%
2760 B%=0
2770 FOR K=1 TO NUM(HTR%)
2780    KK=BETW%(HTL%,K)
2790    AKK%=A%(KK)
2800    FOR I=STARTY% TO P3Y%
2810       FOR J=STARTX% TO WID%
2820          PLAN%(I,J)=KK
2830          B%=B%+1
2840          IF B% )= AKK% GOTO 2870
2850       NEXT J
2860    NEXT I
2870    IF J=FINX% THEN 2950
2880    IF K=NUM(HTR%) THEN KK=BETW%(HTR%,K) ELSE KK=BETW%(HTR%,K+1)
2890    B%=0
2900    FOR L=J+1 TO FINX%
2910       PLAN%(I,J)=KK
2920       B%=B%+1
2930    NEXT L
2940    IF K=NUM(HTR%) THEN 2970
2950    STARTY%=I+1
2960 NEXT K
2970 GOTO 3240
2980 REM TAIL HTR IS ADJACENT TO TOP
2990 STARTX%=P2X%+1
3000 B%=0
3010 K=BETW%(HTR%,1)
3020 FOR I=P1Y% TO P3Y%
3030    FOR J=STARTX% TO WID%
3040       PLAN%(I,J)=K
3050       B%=B%+1
3060       IF B% )= SUMTR% GOTO 3090
3070    NEXT J
3080 NEXT I
3090 IF NUM(HTR%)=1 GOTO 3240
3100 SUMTR%=SUMTR%-A%(BETW%(HTL%,1))
3110 FOR K=2 TO NUM(HTR%)
3120    KK=BETW%(HTR%,K)
3130    STARTX%=STARTX%+1
3140    B%=0
3150    FOR I=P1Y% TO P3Y%
3160       FOR J=STARTX% TO WID%
3170          PLAN%(I,J)=KK
3180          B%=B%+1
3190          IF B% )= SUMTR% THEN 3220
3200       NEXT J
3210    NEXT I
3220    SUMTR%=SUMTR%-B%
3230 NEXT K
3240 REM FILL IN BETWEEN LEFT AND RIGHT
3250 IF NUM(HLR%)=0 GOTO 3770
3260 IF NUM(HLR%)=1 GOTO 3520
```

```
3270 IF SOLUTION%(BETW%(HLR%,2),LEFT)=1 GOTO 3510
3280 STARTX%=P1X%
3290 STARTY%=P2Y%+1
3300 B%=0
3310 FOR K=1 TO NUM(HLR%)
3320    KK=BETW%(HLR%,K)
3330    AKK%=A%(KK)
3340    FOR I=STARTX% TO WID%
3350       FOR J=STARTY% TO P3Y%
3360          PLAN%(J,I)=KK
3370          B%=B%+1
3380          IF B% )= AKK% THEN 3410
3390       NEXT J
3400    NEXT I
3410    IF J=P3Y% THEN 3470
3420    IF K=NUM(HLR%) THEN KK=BETW%(HLR%,K) ELSE KK=BETW%(HLR%,K+1)
3430    B%=0
3440    FOR L=J+1 TO P3Y%
3450       PLAN%(L,I)=KK
3460    NEXT L
3470    IF K=NUM(HLR%) THEN 3500
3480    STARTX%=I+1
3490 NEXT K
3500 GOTO 3770
3510 REM TAIL HLR IS ADJACENT TO LEFT
3520 STARTY%=P2Y%+1
3530 B%=0
3540 K=BETW%(HLR%,1)
3550 FOR I=P1X% TO WID%
3560    FOR J=STARTY% TO P3Y%
3570       PLAN%(J,I)=K
3580       B%=B%+1
3590       IF B% )= SUMLR% THEN 3620
3600    NEXT J
3610 NEXT I
3620 IF NUM(HLR%)=1 GOTO 3770
3630 SUMLR%=SUMLR%-A%(BETW%(HLR%,1))
3640 FOR K=2 TO NUM(HLR%)
3650    KK=BETW%(HLR%,K)
3660    STARTY%=STARTY%+1
3670    B%=0
3680    FOR I=P1X% TO WID%
3690       FOR J=STARTY% TO P3Y%
3700          PLAN%(J,I)=KK
3710          B%=B%+1
3720          IF B% )= SUMLR% THEN 3750
3730       NEXT J
3740    NEXT I
3750    SUMLR%=SUMLR%-B%
3760 NEXT K
3770 REM FILL IN INNER SANCTUM (DISTANCE CLASS 2)
3780 IF CARD%(MAXDIST)=0 GOTO 6040
3790 OM%=ORIGIN%(MAXDIST)
3800 IF CARD%(MAXDIST) ) 1 GOTO 3870
3810 FOR I=P1Y% TO P2Y%-1
```

```
3820    FOR J=P1X% TO P2X%-1
3830       PLAN%(I,J)=OM%
3840    NEXT J
3850 NEXT I
3860 GOTO 6040
3870 IF FUNDTRI%(MAXDIST)=1 GOTO 4820
3880 IF AND%(OM%) ) 1 GOTO 4200
3890 REM OTHERWISE HAVE ONLY A PATH
3900 B%=0
3910 K=OM%
3920 STARTY%=P1Y%
3930 WHILE K () 0
3940    FOR I=STARTY% TO P2Y%
3950       FOR J=P1X% TO P2X%-1
3960          PLAN%(I,J)=K
3970          B%=B%+1
3980          IF B% )= A%(K) GOTO 4010
3990       NEXT J
4000    NEXT I
4010    IF AFTER%(K,1) () 0 THEN KK=AFTER%(K,1) ELSE KK=K:SAVEK%=K
4020    IF J=P2X%-1 GOTO 4080
4030    B%=0
4040    FOR L=J+1 TO P2X%-1
4050       PLAN%(I,L)=KK
4060       B%=B%+1
4070    NEXT L
4080    K=AFTER%(K,1)
4090    IF K=0 GOTO 4120
4100    STARTY%=I+1
4110 WEND
4120 REM CHECK THAT ALL THE INNER SANCTUM IS FILLED
4130 IF I=P2Y% GOTO 6040
4140 FOR L=I+1 TO P2Y%
4150    FOR LK=P1X% TO P2X%-1
4160       PLAN%(L,LK)=SAVEK%
4170    NEXT LK
4180 NEXT L
4190 GOTO 6040
4200 IF AND%(OM%) ) 2 GOTO 4430
4210 REM CHECK WHICH TAIL IS ADJACENT TO LEFT AND RIGHT
4220 REM (ONE OF THEM MUST BE, BY DEFINITION OF TOP, LEFT, RIGHT)
4230 IF ((SOLUTION%(LEFT,AFTER%(OM%,1))=1) AND
        (SOLUTION%(RIGHT%,AFTER%(OM%,1))=1)) GOTO 4260
4240 LL=2
4250 GOTO 4270
4260 LL=1
4270 ML=LL MOD 2 + 1
4280 REM TAIL LL IS ADJACENT TO LEFT AND RIGHT
4290 REM TAIL ML IS ADJACENT TO EITHER LEFT, TOP OR RIGHT,TOP
4300 REM IN ORDER TO FILL IN ORIGIN(MAXDIST), MUST INITIALLY
4310 REM INFLATE ITS AREA BY THE AREA OF TAIL ML
4320 K=AFTER%(OM%,ML)
4330 SUMML%=0
4340 WHILE K () 0
4350    SUMML%=SUMML%+A%(K)
```

```
4360    K=AFTER%(K,1)
4370 WEND
4380 INFLAREA=SUMML%+A%(OM%)
4390 GOSUB 6230
4400 IF SOLUTION%(AFTER%(OM%,ML),LEFT)=1 THEN GOSUB 6640 ELSE GOSUB 6380
4410 GOSUB 6900
4420 GOTO 6040
4430 REM AND(OM)=3
4440 REM CHECK WHICH TAIL IS ADJACENT TO LEFT AND RIGHT
4450 FOR I=1 TO 3
4460    IF ((SOLUTION%(LEFT,AFTER%(OM%,I))=1) AND
            (SOLUTION%(RIGHT%,AFTER%(OM%,I))=1)) GOTO 4480
4470 NEXT I
4480 LL=I
4490 REM CHECK WHICH TAIL IS ADJACENT TO TOP,LEFT
4500 FOR I=1 TO 3
4510    IF I=LL GOTO 4530
4520    IF ((SOLUTION%(LEFT,AFTER%(OM%,I))=1) AND
           (SOLUTION%(TOP%,AFTER%(OM%,I))=1)) GOTO 4540
4530 NEXT I
4540 KL=I
4550 REM OTHER TAIL IS ADJACENT TO RIGHT,TOP
4560 FOR I=1 TO 3
4570    IF ((I <> LL) AND (I <> KL)) THEN ML=I:GOTO 4590
4580 NEXT I
4590 REM TAIL LL IS ADJACENT TO LEFT, RIGHT
4600 REM TAIL KL IS ADJACENT TO TOP, LEFT
4610 REM TAIL ML IS ADJACENT TO TOP, RIGHT
4620 REM INITIALLY INFLATE AREA OF ORIGIN(MAXDIST) WITH BY THE AREAS
4630 REM OF TAILS KL AND ML
4640 K=AFTER%(OM%,KL)
4650 SUMKL%=0
4660 WHILE K <> 0
4670    SUMKL%=SUMKL%+A%(K)
4680    K=AFTER%(K,1)
4690 WEND
4700 K=AFTER%(OM%,ML)
4710 SUMML%=0
4720 WHILE K <> 0
4730    SUMML%=SUMML%+A%(K)
4740    K=AFTER%(K,1)
4750 WEND
4760 INFLAREA=SUMML%+SUMKL%+A%(OM%)
4770 GOSUB 6230
4780 GOSUB 6640
4790 GOSUB 7220
4800 GOSUB 6900
4810 GOTO 6040
4820 REM FUNDTRI(MAXDIST)=1
4830 REM DETERMINE AREAS OF THE TAILS
4840 FOR I=1 TO ANO%(OM%)
4850    K=AFTER%(OM%,I)
4860    IF K=0 THEN 4880
4870    IF ((SOLUTION%(K,LEFT)=1) AND (SOLUTION%(K,TOP%)=1)) THEN KL=I
       ELSE IF ((SOLUTION%(K,RIGHT%)=1) AND (SOLUTION%(K,TOP%)=1))
```

```
            THEN ML=I
4880 NEXT I
4890 SUMKL%=0
4900 SUMML%=0
4910 IF KL=0 THEN 4970
4920 K=AFTER%(OM%,KL)
4930 WHILE K () 0
4940    SUMKL%=SUMKL%+A%(K)
4950    K=AFTER%(K,1)
4960 WEND
4970 IF ML=0 THEN 5030
4980 K=AFTER%(OM%,ML)
4990 WHILE K () 0
5000    SUMML%=SUMML%+A%(K)
5010    K=AFTER%(K,1)
5020 WEND
5030 INFLAREA=SUMKL%+SUMML%+A%(OM%)
5040 GOSUB 6230
5050 IF ((SUMML% () 0) AND (SUMKL% () 0)) THEN 5110
5060 IF SUMML%=0 GOTO 5080
5070 GOSUB 6380
5080 IF SUMKL%=0 GOTO 5130
5090 GOSUB 6640
5100 GOTO 5130
5110 GOSUB 6640
5120 GOSUB 7220
5130 REM CHECK ON VALUE OF KOUNT(MAXDIST)
5140 STARTY%=SAVEI%+1
5150 IF KOUNT(MAXDIST)=0 GOTO 5330
5160 FOR K=KOUNT(MAXDIST)-1 TO 1 STEP -1
5170    B%=0
5180    FOR I=STARTY% TO P2Y%
5190       FOR J=P1X% TO P2X%-1
5200          PLAN%(I,J)=SAVED%(K)
5210          B%=B%+1
5220          IF B% )= A%(SAVED%(K)) GOTO 5250
5230       NEXT J
5240    NEXT I
5250    IF J=P2X%-1 THEN 5310
5260    IF K=1 THEN KK=SAVED%(K) ELSE KK=SAVED%(K-1)
5270    B%=0
5280    FOR L=J+1 TO P2X%-1
5290       PLAN%(I,L)=KK
5300    NEXT L
5310    STARTY%=I+1
5320 NEXT K
5330 REM FILL IN TOP
5340 IF INTOP=OM% THEN 5480
5350 B%=0
5360 FOR I=STARTY% TO P2Y%
5370    FOR J=P1X% TO P2X%-1
5380       PLAN%(I,J)=INTOP
5390       B%=B%+1
5400       IF B% )= A%(INTOP) GOTO 5430
5410    NEXT J
```

```
5420 NEXT I
5430 IF J=P2X%-1 THEN 5470
5440 FOR L=J+1 TO P2X%-1
5450    PLAN%(I,L)=INTOP
5460 NEXT L
5470 STARTY%=I+1
5480 REM FILL IN INLEFT, INRIGHT ; AREA OF INLEFT MUST BE INFLATED BY
5490 REM AREA OF TAIL LL (NOTING THAT TAIL LL IS NOW DIFFERENT TO THE
5500 REM CASES TREATED ABOVE (WHICH HAD FUNDTRI(MAXDIST)=0))
5510 K=AFTER%(INLEFT,1)
5520 SUMLL%=0
5530 WHILE K () 0
5540    SUMLL%=SUMLL%+A%(K)
5550    K=AFTER%(K,1)
5560 WEND
5570 SUMLL%=SUMLL%+A%(INLEFT)
5580 B%=0
5590 FOR I=P1X% TO P2X%-1
5600    FOR J=STARTY% TO P2Y%-1
5610       PLAN%(J,I)=INLEFT
5620       B%=B%+1
5630       IF B% )= SUMLL% THEN 5660
5640    NEXT J
5650 NEXT I
5660 IF J=P2Y%-1 THEN 5720
5670 FOR L=J+1 TO P2Y%-1
5680    PLAN%(L,I)=INLEFT
5690 NEXT L
5700 SAVEI1%=I
5710 REM FILL IN INRIGHT
5720 STARTX%=I+1
5730 FOR I=STARTX% TO P2X%-1
5740    FOR J=STARTY% TO P2Y%-1
5750       PLAN%(J,I)=INRIGHT
5760    NEXT J
5770 NEXT I
5780 REM FILL IN TAIL FROM INLEFT, IF IT EXISTS
5790 SUMLL%=SUMLL%-A%(INLEFT)
5800 IF SUMLL%=0 THEN 6040
5810 RATIO=SUMLL%/(SUMLL%+A%(INLEFT))
5820 WIDTHX%=RATIO*(SAVEI1%-P1X%)
5830 DEPTH%=SUMLL%/WIDTHX%+1
5840 DELTA%=P2Y%-DEPTH%-SAVEI%-2
5850 IF DELTA% ( 0 THEN DEPTH%=DEPTH%+DELTA%
5860 K=AFTER%(INLEFT,1)
5870 FINX%=P1X%+WIDTHX%-1
5880 IF K=0 THEN 6040
5890    B%=0
5900    FOR I=P2Y%-1 TO P2Y%-DEPTH% STEP -1
5910       FOR J=P1X% TO FINX%
5920          PLAN%(I,J)=K
5930          B%=B%+1
5940          IF B% )= SUMLL% THEN 5970
5950       NEXT J
5960    NEXT I
```

```
5970    SUMLL%=SUMLL%-A%(K)
5980    RATIO=SQR(SUMLL%/(SUMLL%+A%(K)))
5990    FINX%=P1X%+RATIO*WIDTHX%
6000    WIDTHX%=RATIO*WIDTHX%
6010    DEPTH%=DEPTH%-1
6020    K=AFTER%(K,1)
6030 GOTO 5880
6040 REM WRITE OUT PLAN MATRIX
6050 LPRINT CHR$(27) CHR$(56)
6060 LPRINT CHR$(29) CHR$(31)
6070 FOR I=1 TO P3Y%
6080    FOR J=1 TO WID%
6090       LPRINT CHAR$(PLAN%(I,J));
6100    NEXT J
6110    LPRINT
6120 NEXT I
6130 LPRINT
6140 LPRINT
6150 LPRINT "CHARACTER SET CODES:"
6160 LPRINT
6170 FOR I=2 TO N
6180    LPRINT "FACILITY" I "IS REPRESENTED BY " CHAR$(I)
6190 NEXT I
6200 LPRINT CHR$(27) CHR$(54)
6210 LPRINT CHR$(30)
6220 END
6230 B%=0
6240 FOR I=P1Y% TO P2Y%-1
6250    FOR J=P1X% TO P2X%-1
6260       PLAN%(I,J)=OM%
6270       B%=B%+1
6280       IF B% )= INFLAREA GOTO 6310
6290    NEXT J
6300 NEXT I
6310 REM MAKE THE BOTTOM OF OM A STRAIGHT LINE
6320 IF J=P2X%-1 GOTO 6360
6330 FOR L=J+1 TO P2X%-1
6340    PLAN%(I,L)=OM%
6350 NEXT L
6360 SAVEI%=I
6370 RETURN
6380 REM FILL IN TAIL ML
6390 K=AFTER%(OM%,ML)
6400 REM ALLOW PROPORTIONAL AREA FOR TAIL ML
6410 RATIO=SUMML%/INFLAREA
6420 WIDTHX%=RATIO*(P2X%-P1X%)
6430 DEPTH%=SUMML%/WIDTHX%+1
6440 DELTA%=DEPTH%+P1Y%-SAVEI%+1
6450 IF DELTA% ) 0 THEN DEPTH%=DEPTH%-DELTA%
6460 FINX%=P2X%-WIDTHX%+1
6470 IF K=0 THEN 6630
6480    B%=0
6490    FOR I=P1Y% TO DEPTH%+P1Y%
6500       FOR J=P2X%-1 TO FINX% STEP -1
6510          PLAN%(I,J)=K
```

```
6520          B%=B%+1
6530          IF B% )= SUMML% GOTO 6560
6540       NEXT J
6550    NEXT I
6560    SUMML%=SUMML%-A%(K)
6570    RATIO=SUMML%/(SUMML%+A%(K))
6580    FINX%=P2X%-RATIO*WIDTHX%+1
6590    WIDTHX%=RATIO*WIDTHX%
6600    DEPTH%=DEPTH%-1
6610    K=AFTER%(K,1)
6620 GOTO 6470
6630 RETURN
6640 REM FILL IN TAIL KL
6650 K=AFTER%(OM%,KL)
6660 RATIO=SUMKL%/INFLAREA
6670 WIDTHX%=RATIO*(P2X%-P1X%)
6680 X%=WIDTHX%
6690 DEPTH%=SUMKL%/WIDTHX%+1
6700 DELTA%=DEPTH%+P1Y%-SAVEI%+1
6710 IF DELTA% ) 0 THEN DEPTH%=DEPTH%-DELTA%
6720 FINX%=P1X%+WIDTHX%-1
6730 IF K=0 THEN 6890
6740    B%=0
6750    FOR I=P1Y% TO DEPTH%+P1Y%
6760       FOR J=P1X% TO FINX%
6770          PLAN%(I,J)=K
6780          B%=B%+1
6790          IF B% )= SUMKL% GOTO 6820
6800       NEXT J
6810    NEXT I
6820    SUMKL%=SUMKL%-A%(K)
6830    RATIO=SUMKL%/(SUMKL%+A%(K))
6840    FINX%=RATIO*WIDTHX%+P1X%
6850    WIDTHX%=RATIO*WIDTHX%
6860    DEPTH%=DEPTH%-1
6870    K=AFTER%(K,1)
6880    GOTO 6730
6890 RETURN
6900 REM FILL IN TAIL LL
6910 REM FIND LAST ELEMENT OF TAIL LL
6920 K=AFTER%(OM%,LL)
6930 WHILE K () 0
6940    SAVEK%=K
6950    K=AFTER%(K,1)
6960 WEND
6970 K=AFTER%(OM%,LL)
6980 B%=0
6990 STARTY%=SAVEI%+1
7000 IF K=SAVEK% THEN 7150
7010    FOR I=STARTY% TO P2Y%-1
7020       FOR J=P1X% TO P2X%-1
7030          PLAN%(I,J)=K
7040          B%=B%+1
7050          IF B% )= A%(K) GOTO 7080
7060       NEXT J
```

```
7070    NEXT I
7080    K=AFTER%(K,1)
7090    IF J=P2X%-1 THEN 7130
7100    FOR L=J+1 TO P2X%-1
7110       PLAN%(I,L)=K
7120    NEXT L
7130    STARTY%=I+1
7140 GOTO 7000
7150 REM FILL IN LAST ELEMENT OF TAIL LL
7160 FOR I=STARTY% TO P2Y%-1
7170    FOR J=P1X% TO P2X%-1
7180       PLAN%(I,J)=SAVEK%
7190    NEXT J
7200 NEXT I
7210 RETURN
7220 K=AFTER%(OM%,ML)
7230 WIDTHX%=P2X%-P1X%-X%-2
7240 DEPTH%=SUMML%/WIDTHX%+1
7250 DELTA%=DEPTH%+P1Y%-SAVEI%+1
7260 IF DELTA% ) 0 THEN DEPTH%=DEPTH%-DELTA%
7270 FINX%=P2X%-WIDTHX%-1
7280 IF K=0 THEN 7440
7290    B%=0
7300    FOR I=P1Y% TO DEPTH%+P1Y%
7310       FOR J=P2X%-1 TO FINX% STEP -1
7320          PLAN%(I,J)=K
7330          B%=B%+1
7340          IF B% )= SUMML% GOTO 7370
7350       NEXT J
7360    NEXT I
7370    SUMML%=SUMML%-A%(K)
7380    RATIO=SUMML%/(SUMML%+A%(K))
7390    FINX%=P2X%-RATIO*WIDTHX%-2
7400    WIDTHX%=RATIO*WIDTHX%
7410    DEPTH%=DEPTH%-1
7420    K=AFTER%(K,1)
7430    GOTO 7280
7440 RETURN
```